Logowanie do PostgreSQL kontem z Active Directory: LDAP, LDAPS i grupy

Poziom: średnio **Dotyczy: PostgreSQL 16 na pg01, Active Directory ad.zielona.example**

Czas: ~35 min

PostgreSQL potrafi sprawdzić hasło nie u siebie, tylko w Active Directory, więc programista loguje się do bazy tym samym hasłem co do komputera. Strona pokazuje oba tryby metody `ldap` (simple bind i search+bind), szyfrowanie LDAPS, konto techniczne, zaufanie do certyfikatu `CA-Zielona`, związek grup z AD z rolami w bazie i dostęp czasowy przez członkostwo w grupie. Wracamy tu przy każdej nowej bazie, do której ma się logować ktoś z zespołu Oli.

ZAŁOŻENIA SCENARIUSZA

kontrolery `dc01` i `dc02` mają certyfikat LDAPS wystawiony przez urząd `CA-Zielona`, a `pg01` ma włączone `ssl = on` z własnym certyfikatem, tak jak opisuje strona Uwierzytelnianie i bezpieczeństwo połączeń. W lesie `ad.zielona.example` działa funkcja Privileged Access Management (decyzja autora z 11 października 2026), więc członkostwo w grupie może mieć czas życia. Programiści łączą się z sieci biura 10.20.20.0/24.

Po przenosinach urzędu według strony Przenosiny urzędu z dc01 bez przerwy: dwa urzędy równolegle i wygaszenie CA-Zielona dokładamy tu w ten sam sposób zaufanie do `Zielona-Root-CA`, a `CA-Zielona.crt` zostaje obok do wygaszenia starego urzędu.

Komendy `ldapsearch`, `openssl` i `psql` wykonujemy w CLI bash na `pg01` jako użytkownik z uprawnieniami sudo, komendy SQL w `psql` kontem `postgres`, a komendę PowerShell na stacji administratora z modułem ActiveDirectory.

Skróty użyte na tej stronie (12)

Skrót	Co znaczy

AD	Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
DN	Distinguished Name: pełny, jednoznaczny adres jednego obiektu w katalogu LDAP, na przykład konta albo grupy
GLPI	Gestionnaire Libre de Parc Informatique: otwarty system do zgłoszeń, inwentarza i bazy wiedzy IT; w Zielonej działa w kontenerze na docker01
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
LDAPS	LDAP szyfrowany TLS, zwykle port 636
PAM	Pluggable Authentication Modules: wymienne moduły logowania w Linuksie, decydują na przykład o sile wymaganego hasła; w Active Directory także Privileged Access Management: funkcja lasu pozwalająca nadawać członkostwo w grupie na określony czas
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
SSL	Secure Sockets Layer: dawna nazwa szyfrowania, dziś formalnie TLS
StartTLS	Start Transport Layer Security: polecenie, które zamienia zwykłe, nieszyfrowane połączenie (np. LDAP na porcie 389) w szyfrowane już po jego nawiązaniu
UPN	User Principal Name: login w formie adresu, np. jan.nowak@zielona.example

Fiszka

TYLKO ODCZYT

Sprawdzenie certyfikatu LDAPS kontrolera domeny: openssl s_client

```
openssl s_client -connect NazwaKontroleraDC:636 -CAfile /etc/pki/tls/certs/ca-bundle.crt
-verify_return_error </dev/null | grep "Verify return code"
```

> (gdzie `NazwaKontroleraDC` to pełna nazwa kontrolera, np. `dc01.ad.zielona.example`; nazwa musi być w certyfikacie, więc nie podajemy adresu IP)

Pierwszy test przed dotknięciem `pg_hba.conf`: czy port 636 odpowiada i czy `pg01` ufa certyfikatowi kontrolera. Wynik inny niż `0 (ok)` oznacza, że PostgreSQL też się nie połączy.

Przykład:

```
Verify return code: 0 (ok)
```

LDAPS = LDAP szyfrowany TLS, zwykle port 636 · **IP** = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection) · **openssl** = zestaw narzędzi do certyfikatów i szyfrowania

ZMIENIA SYSTEM

Dodanie certyfikatu CA-Zielona do zaufanych na pg01: update-ca-trust

```
sudo cp CA-Zielona.crt /etc/pki/ca-trust/source/anchors/
sudo update-ca-trust
```

Rocky Linux 9 dokłada certyfikat urzędu do wspólnego pliku `/etc/pki/tls/certs/ca-bundle.crt`, z którego korzystają `openssl`, `ldapsearch` i PostgreSQL. Certyfikat `CA-Zielona.crt` eksportujemy z `dc01` w formacie Base64, tak jak przy innych aplikacjach z LDAPS.

LDAPS = LDAP szyfrowany TLS, zwykle port 636 · **cp** = copy: skopiuj · **update-ca-trust** = update CA trust: odświeżenie systemowego magazynu zaufanych urzędów certyfikacji w Rocky Linux

TYLKO ODCZYT

Wskazanie pliku urzędów dla biblioteki OpenLDAP: /etc/openldap/ldap.conf

```
TLS_CACERT      /etc/pki/tls/certs/ca-bundle.crt
TLS_REQCERT     demand
```

PostgreSQL rozmawia z AD przez bibliotekę OpenLDAP, a ta czyta zaufane urzędy z `TLS_CACERT` [4]. `TLS_REQCERT demand` to wartość domyślna (połączenie bez poprawnego certyfikatu jest przerywane), wpisujemy ją jawnie, żeby nikt nie zmienił jej na `never` „na chwilę”.

AD = Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie

ZMIENIA SYSTEM

Utworzenie konta technicznego do wyszukiwania: New-ADUser

```
New-ADUser -Name "svc_pg01_ldap" -SamAccountName "svc_pg01_ldap" -Path
"OU=KontaTechniczne,DC=ad,DC=zielona,DC=example" -AccountPassword (Read-Host -
AsSecureString "Haslo") -Enabled $true -CannotChangePassword $true -Description
"PostgreSQL pg01: wyszukiwanie kont, tylko odczyt"
```

Konto zwykłego użytkownika domeny bez żadnych dodatkowych grup: domyślne prawo odczytu katalogu wystarcza do wyszukania konta i jego grup. Hasło od razu trafia do Vaultwardena, bo za chwilę wpiszemy je do `pg_hba.conf`.

OU = Organizational Unit: folder w Active Directory grupujący konta i komputery · **DC** = Domain Controller: serwer, który trzyma Active Directory i loguje użytkowników

TYLKO ODCZYT

Wyszukanie konta tak, jak zrobi to PostgreSQL: ldapsearch

```
ldapsearch -H ldaps://NazwaKontroleraDC -D
"CN=svc_pg01_ldap,OU=KontaTechniczne,DC=ad,DC=zielona,DC=example" -W -b
"DC=ad,DC=zielona,DC=example" "(&(sAMAccountName=nazwa_usera)(memberOf=DNGrupy))" dn
```

> (gdzie `nazwa_usera` to login domenowy, a `DNGrupy` to pełna nazwa grupy, np. `CN=GG_Baza_Cennik_RO,OU=Grupy,DC=ad,DC=zielona,DC=example`)

To ten sam bind i ten sam filtr, który wpiszemy do `pg_hba.conf`. Dokładnie jeden wpis `dn:` oznacza, że PostgreSQL znajdzie użytkownika; zero wpisów oznacza, że osoby nie ma w grupie albo filtr ma literówkę.

Przykład (fragment):

```
dn: CN=Ola Wisniewska,OU=Uzytkownicy,DC=ad,DC=zielona,DC=example
# numEntries: 1
```

CN = Common Name: główna nazwa w certyfikacie, np. adres serwera · **OU** = Organizational Unit: folder w Active Directory grupujący konta i komputery · **DC** = Domain Controller: serwer, który trzyma Active Directory i loguje użytkowników · **ldapsearch** = LDAP search: wyszukuje wpisy w katalogu LDAP (np. w Active Directory) według filtra, tu do sprawdzenia, czy konto przechodzi filtr z pg_hba.conf

TYLKO ODCZYT

Reguła search+bind przez LDAPS: pg_hba.conf

```
hostssl  cennik  +cennik_ro  10.20.20.0/24  ldap \
    ldapserver="dc01.ad.zielona.example dc02.ad.zielona.example" ldapscheme=ldaps \
    ldapbasedn="DC=ad,DC=zielona,DC=example" \
    ldapbinddn="CN=svc_pg01_ldap,OU=KontaTechniczne,DC=ad,DC=zielona,DC=example" \
    ldapbindpasswd="HasloKontaTechnicznego" \

ldapsearchfilter="(&(sAMAccountName=$username)(memberOf=CN=GG_Baza_Cennik_RO,OU=Grupy,DC
=ad,DC=zielona,DC=example))"
```

Jedna reguła w pięciu liniach: PostgreSQL 16 łączy linie zakończone ukośnikiem wstecznym [1]. `hostssl` wymusza szyfrowanie między klientem a `pg01`, `ldapscheme=ldaps` między `pg01` a kontrolerem, a `+cennik_ro` ogranicza regułę do ról należących do roli grupowej `cennik_ro`. Regułę wstawiamy nad szerszymi regułami dla tej sieci.

LDAPS = LDAP szyfrowany TLS, zwykle port 636

ZMIENIA SYSTEM

Utworzenie roli grupowej i roli dla konta domenowego: CREATE ROLE

```
CREATE ROLE cennik_ro NOLOGIN;
GRANT CONNECT ON DATABASE cennik TO cennik_ro;
GRANT USAGE ON SCHEMA public TO cennik_ro;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO cennik_ro;
CREATE ROLE "nazwa_usera" LOGIN IN ROLE cennik_ro;
```

> (gdzie `nazwa_usera` to login domenowy małymi literami, w cudzysłowie, bo zawiera kropkę, np. `"ola.wisniewska"`)

Rola konta domenowego nie ma hasła, bo hasło sprawdza AD. Prawa dostaje przez rolę grupową, więc przy nowej osobie dodajemy tylko jedną rolę, a nie powtarzamy `GRANT`.

AD = Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie

TYLKO ODCZYT

Sprawdzenie, jak silnik odczytał regułę: pg_hba_file_rules

```
SELECT line_number, type, database, user_name, auth_method, error
FROM pg_hba_file_rules
WHERE auth_method = 'ldap' OR error IS NOT NULL;
```

Sprawdzamy przed przeładowaniem: pusta kolumna `error` przy regule `ldap` znaczy, że składnia i nazwy opcji są poprawne. Widok domyślnie czyta tylko superużytkownik [1].

Przykład:

```
 line_number |  type   | database |  user_name   | auth_method | error
-------------+---------+----------+--------------+-------------+-------
          12 | hostssl | {cennik} | {+cennik_ro} | ldap        |
(1 row)
```

□□ TYLKO ODCZYT

Przeładowanie reguł: pg_reload_conf()

```
SELECT pg_reload_conf();
```

Reguły z `pg_hba.conf` wchodzą w życie po przeładowaniu, bez restartu; otwarte sesje działają dalej po staremu.

□□ TYLKO ODCZYT

Test logowania kontem domenowym: psql

```
psql "host=pg01.ad.zielona.example dbname=cennik user=nazwa_usera sslmode=verify-full"
```

Pyta o hasło domenowe. Udane logowanie potwierdza cały łańcuch: regułę, LDAPS, konto techniczne, filtr z grupą i rolę w bazie.

□□ **LDAPS** = LDAP szyfrowany TLS, zwykle port 636 · **psql** = PostgreSQL SQL: klient bazy PostgreSQL

□□ TYLKO ODCZYT

Zakończenie sesji po końcu dostępu czasowego: pg_terminate_backend

```
SELECT pg_terminate_backend(pid) FROM pg_stat_activity WHERE usename = 'nazwa_usera' AND datname = 'cennik';
```

PostgreSQL sprawdza grupę tylko przy logowaniu, więc sesja otwarta przed wygaśnięciem członkostwa trwa do rozłączenia. W dniu końca dostępu kończymy ją sami, jeśli wciąż wisi.

Z życia wzięte

Z ŻYCIA WZIĘTE

Środa, 9:40. Ola dostała w poniedziałek dostęp do bazy `cennik` na tydzień, a dziś `psql` odpowiada jej `FATAL: LDAP authentication failed for user "ola.wisniewska"`. W logu `pg01` stoi `LDAP user "ola.wisniewska" does not exist`, choć konto w AD działa. Tomek wkleja filtr z `pg_hba.conf` do `ldapsearch` i dostaje zero wpisów: we wtorek ktoś przeniósł grupę `GG_Baza_Cennik_RO` do innej jednostki, więc jej pełna nazwa w filtrze przestała pasować. Po poprawce DN w regule i przeładowaniu Ola loguje się o 9:55, a do standardu trafia zasada, że grup użytych w filtrach baz nie przenosimy bez zmiany w GLPI.

Jak to działa

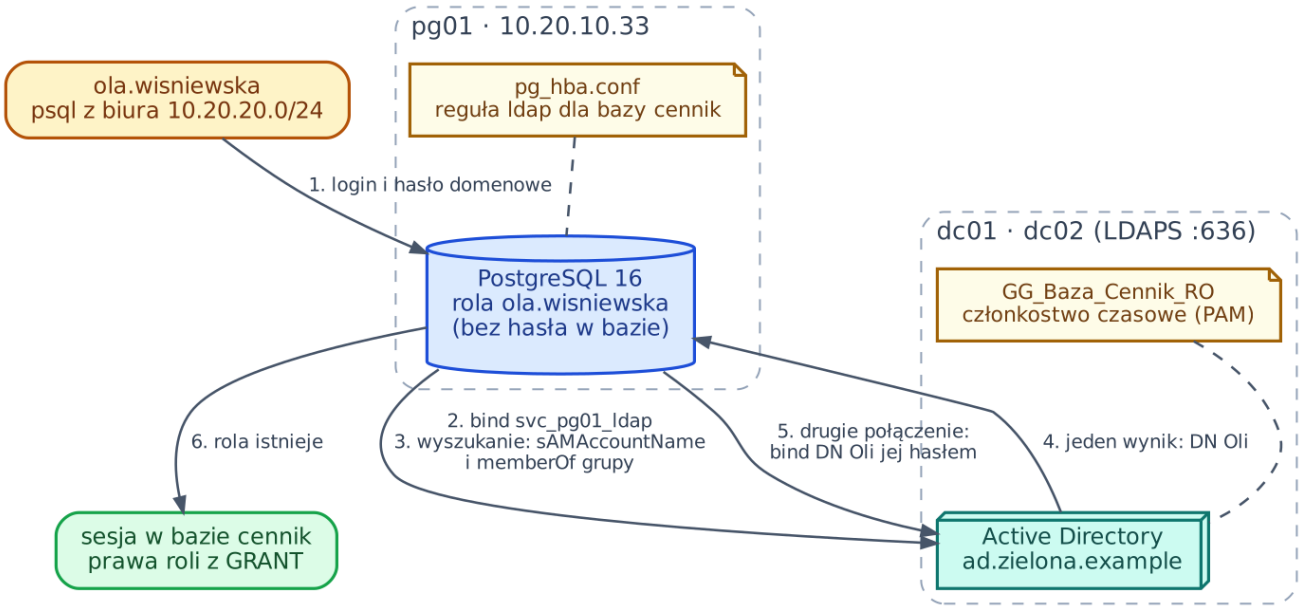
Rozwiń: jak to działa pod spodem

Hasło sprawdza AD, rola zostaje w bazie

Pojęcie: metoda ldap. Metoda uwierzytelniania w `pg_hba.conf`, która działa jak zwykłe hasło, tylko parę login i hasło sprawdza serwer katalogu, a nie PostgreSQL [2]. LDAP służy wyłącznie do sprawdzenia hasła, więc rola o tej samej nazwie musi już istnieć w bazie.

> **Pojęcie: bind.** Zalogowanie się do serwera LDAP: klient podaje pełną nazwę obiektu (DN) i hasło, a serwer odpowiada, czy pasują.

Klient wysyła hasło do PostgreSQL otwartym tekstem, tak jak w metodzie `password`. Dlatego reguła stoi na `hostssl`, a nie `host`: `ldapscheme` szyfruje tylko odcinek od `pg01` do kontrolera, a odcinek od klienta do `pg01` szyfruje wyłącznie SSL samego PostgreSQL [2].



Rysunek 1. Logowanie w trybie search+bind: Ola podaje hasło, pg01 wiązaniem konta svc_pg01_ldap wyszukuje jej konto z filtrem na grupę, a potem drugim połączeniem sprawdza jej hasło w AD (kliknięcie otwiera pełny rozmiar)

Dwa tryby: simple bind i search+bind

	Simple bind	Search+bind
Co robi PostgreSQL	składa DN z przedrostka, loginu i przyrostka i od razu wiąże się jako użytkownik	wiąże się kontem technicznym (albo anonimowo), wyszukuje użytkownika, potem wiąże się jako znaleziony użytkownik jego hasłem
Opcje	`ldapprefix`, `ldapsuffix`	`ldapbasedn`, `ldapbinddn`, `ldapbindpasswd`, `ldapsearchattribute` albo `ldapsearchfilter`

	Simple bind	Search+bind
Konto techniczne	niepotrzebne	potrzebne, bo AD nie pozwala na wyszukiwanie anonimowe
Sprawdzenie grupy	niemożliwe: AD sprawdza tylko hasło	filtr z `memberOf`
Gdy konto jest w innej jednostce	działa, jeśli DN da się złożyć z loginu	działa, wyszukiwanie obejmuje całe poddrzewo

Opcje wspólne dla obu trybów: `ldapserver` (kilka nazw rozdzielonych spacją), `ldapport` (bez niego port domyślny biblioteki), `ldapscheme=ldaps`, `ldaptls=1` (StartTLS) i `ldapurl`. Mieszanie opcji simple bind z opcjami search+bind w jednej regule jest błędem [2].

W AD simple bind działa z loginem w postaci UPN, gdy UPN kont kończy się na `@ad.zielona.example` (sprawdzamy w AD, zanim tak skonfigurujemy):

```
hostssl  NazwaBazy  all  10.20.20.0/24  ldap ldapserver=dc01.ad.zielona.example
ldapscheme=ldaps ldapsuffix="@ad.zielona.example"
```

Tak zalogować się może każdy pracownik, który ma rolę w bazie, bez względu na grupy. W Zielonej używamy search+bind, bo cały pomysł dostępu na wniosek opiera się na grupie.

> **Pojęcie: StartTLS.** Polecenie, które zamienia zwykłe połączenie LDAP na porcie 389 w szyfrowane, już po jego nawiązaniu. LDAPS na porcie 636 szyfruje od pierwszego bajtu. PostgreSQL zna oba: `ldaptls=1` albo `ldapscheme=ldaps`; dokumentacja nazywa LDAPS sposobem niestandardowym, ale AD go obsługuje i reszta bazy (Gitea, Jenkins, Zabbix) też łączy się przez 636 [2][5].

Certyfikat i zaufanie

Kontroler domeny wystawia LDAPS tylko wtedy, gdy ma certyfikat z rozszerzeniem Server Authentication, z pełną nazwą kontrolera w polu podmiotu albo w nazwach alternatywnych, wydany przez urząd, któremu ufają kontroler i klienci [5]. Stąd dwie zasady: w `ldapserver` stoją pełne nazwy `dc01.ad.zielona.example`, a nie adresy IP, i `pg01` musi ufać `CA-Zielona`.

PostgreSQL nie ma własnej opcji z plikiem urzędu dla LDAP. Zaufanie bierze z konfiguracji biblioteki OpenLDAP, czyli z `/etc/openldap/ldap.conf` [4]. Gdy `ldapsearch` z konsoli działa, a PostgreSQL nie, najpierw sprawdzamy, czy usługa `postgresql-16` nie ma w środowisku innego `LDAPCONF` albo `LDAPTLS_CACERT`.

Konto techniczne

`svc_pg01_ldap` służy tylko do wyszukiwania. Nie należy do żadnej grupy poza Domain Users, nie loguje się na komputery, a jego hasło stoi jawnym tekstem w `pg_hba.conf`. Dlatego:

- plik `pg_hba.conf` czyta tylko użytkownik `postgres` (sprawdzamy `ls -l` w katalogu danych),
- kopia katalogu danych w pgBackRest zawiera ten plik, więc repozytorium kopii jest tak samo tajne jak hasło,
- przy zmianie hasła konta podmieniamy je w regule i w Vaultwardenie w jednej zmianie, inaczej po wygaśnięciu starego hasła nikt nie zaloguje się do `cennik`.

Filtr z grupą i grupy zagnieżdżone

`ldapsearchfilter` to filtr LDAP, w którym `$username` zamienia się na login podany przez klienta; `ldapsearchattribute=sAMAccountName` byłby równoważny filtrowi `(sAMAccountName=$username)` [2]. Dokładamy do niego warunek `memberOf`, więc wyszukiwanie zwraca konto tylko członka grupy.

Zwykły warunek `memberOf=DN` sprawdza członkostwo bezpośrednie. Gdy do grupy dostępu dodajemy inne grupy, potrzebna jest reguła dopasowania `LDAP_MATCHING_RULE_IN_CHAIN` (identyfikator 1.2.840.113556.1.4.1941), która przechodzi cały łańcuch grup [6]:

```
ldapsearchfilter="(&(sAMAccountName=$username)(memberOf:1.2.840.113556.1.4.1941:=CN=GG_Baz
a_Cennik_RO,OU=Grupy,DC=ad,DC=zielona,DC=example))"
```

Microsoft ostrzega, że takie zapytania mocniej obciążają kontroler [6]. W Zielonej grupy dostępu do baz zawierają ludzi, nie inne grupy, więc zostajemy przy zwykłym `memberOf`.

Grupy w AD a role w bazie

PostgreSQL nie zakłada ról sam i nie synchronizuje ich z AD: grupa w AD decyduje, kto w ogóle przejdzie logowanie, a rola i jej `GRANT` w bazie decydują, co ta osoba zobaczy. Obie warstwy utrzymujemy razem.

W AD	W PostgreSQL	Kto zmienia
grupa `GG_Baza_Cennik_RO`	rola grupowa `cennik_ro` z `GRANT SELECT`	my, raz, przy nowej bazie
członkostwo osoby w grupie, czasowe albo stałe	rola `"ola.wisniewska"` z `LOGIN IN ROLE cennik_ro`, na stałe	członkostwo: dyżurny z wniosku KU-05; rola: przy pierwszym wniosku danej osoby

W AD	W PostgreSQL	Kto zmienia
grupa zapisu (np. `GG_Baza_Cennik_RW`)	osobna rola `cennik_rw`, nadawana `GRANT cennik_rw TO` na czas wniosku	dyżurny, z `REVOKE` w dniu końca

Zapis wymaga obu kroków, bo prawa w bazie wynikają z roli, a nie z grupy w AD. Filtr reguły musi wtedy przepuszczać członków obu grup, na przykład `(|(memberOf=DN grupy RO)(memberOf=DN grupy RW))` wewnątrz warunku z `sAMAccountName`.

Gdy osób i baz robi się dużo, ręczne zakładanie ról męczy. Istnieją narzędzia, które czytają grupy z LDAP i zakładają albo usuwają role według reguł, na przykład `pg-ldap-sync` (Ruby) i `ldap2pg` (Dalibo). Podajemy je informacyjnie: w Zielonej ich nie wdrażamy, a przed ewentualnym wdrożeniem sprawdzamy w ich dokumentacji obsługiwane wersje PostgreSQL.

Dostęp czasowy z Privileged Access Management

> **Pojęcie: Privileged Access Management (PAM) w AD.** Funkcja lasu Active Directory od poziomu Windows Server 2016, która pozwala dodać konto do grupy z czasem życia; po jego upływie członkostwo samo znika [3]. Skrót PAM ma też drugie znaczenie: Pluggable Authentication Modules w Linuksie, z którymi PostgreSQL ma osobną metodę `pam`. Na tej stronie PAM znaczy zawsze funkcję AD.

Dostęp „na chwilę” do `cennik` to członkostwo czasowe w `GG_Baza_Cennik_RO`, dodawane `Add-ADGroupMember -MemberTimeToLive` według strony Wnioski o usługę, katalog usług i onboarding: od nowego konta do odebrania dostępu. Usunięcie wygasłego członkostwa kontroler ocenia od razu, bez czekania na replikację [3], więc po terminie filtr `memberOf` przestaje przepuszczać tę osobę przy kolejnym logowaniu. Zachowanie w naszej domenie potwierdzamy w dniu końca tym samym `ldapsearch` co w Fiszce: zero wpisów znaczy, że dostęp wygasł.

Trzy rzeczy, których członkostwo czasowe nie załatwia:

1. Sesja otwarta przed terminem trwa do rozłączenia; w dniu końca sprawdzamy `pg_stat_activity` i w razie potrzeby kończymy ją `pg_terminate_backend`.
2. `VALID UNTIL` przy roli dotyczy tylko hasła trzymanego w PostgreSQL, więc przy logowaniu przez LDAP nic nie zmienia; zostaje drogą zapasową dla osób bez konta w domenie.
3. Prawa zapisu nadane `GRANT cennik_rw` nie wygasają same; `REVOKE` jest krokiem zgłoszenia.

Kolejność reguł i brak drugiej szansy

PostgreSQL wybiera pierwszą regułę, która pasuje do typu połączenia, bazy, roli i adresu. Gdy uwierzytelnienie w tej regule się nie uda, kolejnych reguł nie sprawdza [1]. Reguła `ldap` dla `cennik` musi więc stać nad ogólną regułą `scram-sha-256` dla sieci biura, a rola, która ma logować się hasłem w PostgreSQL, nie może należeć do `cennik_ro`, bo trafi w regułę LDAP i tam przepadnie.

Częste błędy

Objaw	Przyczyna	Co robimy
w logu `LDAP user "ola.wisniewska" does not exist`, choć konto w AD działa	wyszukiwanie z filtrem nie zwróciło konta: osoby nie ma w grupie, członkostwo wygasło, grupa zmieniła DN albo `ldapbasedn` nie obejmuje jednostki z kontem	uruchamiamy `ldapsearch` z Fiszki z tym samym filtrem, poprawiamy DN albo dodajemy osobę z wniosku
`could not search LDAP for filter` z diagnostyką `000004DC`	wyszukiwanie bez udanego bindu: najczęściej brak `ldapbinddn`, więc PostgreSQL próbuje anonimowo, a AD nie pozwala na wyszukiwanie anonimowe; możliwe też podążanie biblioteki za odsyłaczami do innych partycji katalogu przy wyszukiwaniu od korzenia domeny	uzupełniamy `ldapbinddn` i `ldapbindpasswd`, sprawdzamy je `ldapsearch -D ... -W`; przy odsyłaczach zawężamy `ldapbasedn` do jednostki z kontami (sprawdzamy w dokumentacji OpenLDAP opcję `REFERRALS` w `ldap.conf`)
logowanie działa z `ldapsearch`, z PostgreSQL nie, w logu błąd połączenia z serwerem LDAP	biblioteka OpenLDAP w usłudze nie ufa certyfikatowi albo w `ldapserver` stoi adres IP	`TLS_CACERT` w `/etc/openldap/ldap.conf`, pełne nazwy kontrolerów, test `openssl s_client`
`FATAL: role "ola.wisniewska" does not exist` po udanym haśle	PostgreSQL nie zakłada ról z AD	`CREATE ROLE "ola.wisniewska" LOGIN IN ROLE cennik_ro` z wniosku
Ola wpisuje `Ola.Wisniewska` i nie wchodzi	AD porównuje loginy bez względu na wielkość liter, PostgreSQL nazwy ról z nią	role zakładamy małymi literami i tak podajemy login w instrukcji dla zespołu
konto Oli zablokowane w AD po porannych próbach aplikacji	złe hasło w pliku połączenia aplikacji, każda próba liczy się do blokady konta w domenie	aplikacje łączą się własnym kontem technicznym w PostgreSQL, nie kontem człowieka
nikt nie loguje się do `cennik`, gdy `dc01` jest w aktualizacji	w `ldapserver` stoi jeden kontroler	dwa kontrolery rozdzielone spacją w cudzysłowie
po zmianie hasła `svc_pg01_ldap` logowanie przestaje działać	stare hasło w `ldapbindpasswd`	nowe hasło w regule, `pg_reload_conf()`, test `psql` w tej samej zmianie

Objaw	Przyczyna	Co robimy
audytor pyta, czemu hasła lecą otwartym tekstem	reguła `host` zamiast `hostssl` przy metodzie `ldap`	`hostssl` i `sslmode=verify-full` po stronie klientów

Ćwiczenia

1. Tomek proponuje uprościć regułę do trybu simple bind z `ldapsuffix="@ad.zielona.example"`, „bo nie trzeba konta technicznego”. Co tracimy?

Odpowiedź

Sprawdzenie grupy. W simple bind AD odpowiada tylko, czy hasło pasuje, więc zaloguje się każdy pracownik z rolą w bazie, także po wygaśnięciu członkostwa czasowego. Dostęp na wniosek KU-05 przestałby działać, bo nic nie łączy logowania z grupą.

1. Wniosek Oli o odczyt `cennik` wygasa w piątek o 16:00. O 16:30 Ola nadal pracuje w otwartym oknie `psql`. Dlaczego i co robimy?

Odpowiedź

PostgreSQL sprawdza grupę tylko przy logowaniu, a sesja otwarta przed terminem trwa do rozłączenia. Dyżurny sprawdza `pg_stat_activity`, kończy sesję `pg_terminate_backend` i zapisuje to w zgłoszeniu. Nowe logowanie już się nie uda, bo filtr `memberOf` nie znajdzie Oli.

1. Ola ma dostać na jeden dzień prawo zapisu w `cennik`. Dyżurny dodaje ją do grupy zapisu w AD z `-MemberTimeToLive`. Czy to wystarczy?

Odpowiedź

Nie. Grupa w AD tylko otwiera logowanie; prawa w bazie wynikają z roli. Potrzebne jest jeszcze `GRANT cennik_rw TO "ola.wisniewska"`, a w dniu końca `REVOKE`, bo nadanie w PostgreSQL samo nie wygasa. Filtr reguły musi też przepuszczać członków grupy zapisu.

1. Po przeniesieniu konta Oli do jednostki `OU=Programisci,DC=ad,DC=zielona,DC=example` logowanie przestało działać, a filtr w `ldapsearch` z bazą `DC=ad,DC=zielona,DC=example` znajduje konto. Gdzie szukamy błędu?

Odpowiedź

W `ldapbasedn` reguły: jeżeli ktoś zawęził go do `OU=Uzytkownicy,...`, wyszukiwanie PostgreSQL nie obejmuje nowej jednostki, choć `ldapsearch` z szerszą bazą konto widzi. Porównujemy opcje z `pg_hba.conf` z poleceniem testowym znak po znaku, najlepiej kopiując je z pliku.

Powiązane: Uwierzytelnianie i bezpieczeństwo połączeń · postgresql.conf i pg_hba.conf · Użytkownicy, role i uprawnienia · Sesje, długie zapytania, kasowanie kwerendy · Wnioski o usługę, katalog usług i onboarding: od nowego konta do odebrania dostępu · Logowanie kontem z Active Directory

Źródła (stan na 19.10.2026):

1. postgresql.org/docs/16: The pg_hba.conf File, łączenie linii ukośnikiem wstecznym, `hostssl`, znak `+` przy roli, brak przejścia do kolejnej reguły po nieudanym uwierzytelnieniu, przeładowanie; pg_hba_file_rules, widok domyślnie tylko dla superużytkownika.
2. postgresql.org/docs/16: LDAP Authentication, tryby simple bind i search+bind, opcje `ldapserver`, `ldapport`, `ldapscheme`, `ldaptls`, `ldapprefix`, `ldapsuffix`, `ldapbasedn`, `ldapbinddn`, `ldapbindpasswd`, `ldapsearchattribute`, `ldapsearchfilter`, `ldapurl`, zakaz mieszania trybów, rola musi istnieć w bazie, szyfrowanie tylko odcinka do serwera LDAP.
3. learn.microsoft.com: What's new in Active Directory Domain Services in Windows Server 2016, członkostwo z czasem życia (expiring links) dla `member` i `memberOf`; Privileged Access Management for Active Directory Domain Services, wygasłe członkostwo oceniane od razu na każdym kontrolerze.
4. man7.org: ldap.conf(5), `TLS_CACERT`, `TLS_REQCERT` z wartością domyślną `demand`, zmienne `LDAPCONF` i `LDAP<opcja>`.
5. learn.microsoft.com: Enable LDAP over SSL with a third-party certification authority, wymagania certyfikatu LDAPS kontrolera i port 636.
6. learn.microsoft.com: Search Filter Syntax, `LDAP_MATCHING_RULE_IN_CHAIN` (1.2.840.113556.1.4.1941) i uwaga o obciążeniu.
7. Komunikaty `LDAP user "..." does not exist`, `LDAP authentication failed for user` i `could not search LDAP for filter` z `000004DC`: wątki na liście pgsql-general (A vexing problem with LDAP, Test connection user with ldaps server). Narzędzia `pg-ldap-sync` i `ldap2pg`: tylko wzmianka, wersji nie sprawdzaliśmy.

Revision #3
Created 2026-10-18 22:59:46 UTC by Tomasz Konieczny
Updated 2026-10-26 14:12:31 UTC by Tomasz Konieczny