DNS z perspektywy administratora: dig, host, resolvectl

◫◫ Poziom: średni

◫◫ Dotyczy: RHEL/Rocky 9–10 (systemd-resolved), różnice Debian/Ubuntu i SLES zaznaczone

▯ Czas: ~20 min

„`dig` pokazuje dobry adres, a przeglądarka i tak się nie łączy" to jeden z najczęstszych zgłoszonych problemów, który wcale nie jest problemem DNS — to problem tego, że `dig` i aplikacja pytają o nazwę zupełnie inną drogą. Ta strona pokazuje obie drogi i narzędzia do sprawdzenia każdej z nich osobno.

◫◫ Skróty użyte na tej stronie (17)

Skrót	Co znaczy
AD	Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie
DHCP	Dynamic Host Configuration Protocol: usługa, która sama rozdaje komputerom adresy IP
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
IPv4	Internet Protocol wersja 4: adresy zapisane czterema liczbami z kropkami, na przykład 10.20.10.77
IPv6	Internet Protocol wersja 6: nowszy, dłuższy zapis adresów, w liczbach szesnastkowych oddzielonych dwukropkami
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
MX	Mail Exchanger: rekord DNS wskazujący serwer poczty domeny
PTR	rekord DNS odwrotny: zamienia adres IP na nazwę
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL

Skrót	Co znaczy
SPF	Sender Policy Framework: rekord DNS mówiący, które serwery mogą wysyłać pocztę domeny
SRV	rekord DNS wskazujący nie adres hosta, tylko usługę i port, pod którym jej szukać, np. kontroler domeny w sieci
SSSD	System Security Services Daemon: usługa na kliencie linuksowym tłumacząca logowanie, sudo i sprawdzanie grup na zapytania do katalogu, czy to FreeIPA, czy Active Directory
TLD	Top Level Domain: najwyższy poziom nazwy domeny internetowej, część po ostatniej kropce, np. `.com` albo `.example`
TTL	Time To Live: czas życia rekordu lub danych
TXT	typ rekordu DNS przechowującego dowolny tekst, wykorzystywany między innymi do publikowania reguł SPF, DKIM i DMARC

Fiszka

□□ TYLKO ODCZYT

Wysłanie pełnego zapytania z licznikiem czasu: dig NazwaDomeny

```
dig NazwaDomeny
```

Wysyła zapytanie DNS bezpośrednio do serwera z `/etc/resolv.conf` i pokazuje pełną odpowiedź: nagłówek, sekcję pytania, sekcję odpowiedzi, czas odpowiedzi i użyty serwer. Domyślny, najbardziej szczegółowy widok — dobry punkt startu, gdy nie wiadomo jeszcze, czego szukać.

Przykład:

```
; <<>> DiG 9.16.23 <<>> app01.zielona.example
;; ANSWER SECTION:
app01.zielona.example.   3600    IN      A       10.20.10.41

;; Query time: 2 msec
;; SERVER: 10.20.10.11#53(10.20.10.11)
```

□□ **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP ·**dig** = domain information groper: odpytywanie DNS

TYLKO ODCZYT

Wypisanie samego wyniku, bez ceremonii: dig +short

```
dig +short NazwaDomeny
```

Wycina cały nagłówek i sekcje diagnostyczne, zostawiając same adresy — wygodne do użycia w skryptach (`IP=$(dig +short nazwa)`).

Przykład:

```
10.20.10.41
```

IP = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection) · **dig** = domain information groper: odpytywanie DNS

TYLKO ODCZYT

Prześledzenie pełnej drogi od korzenia DNS: dig +trace

```
dig +trace NazwaDomeny
```

Zamiast pytać jeden serwer i wierzyć jego odpowiedzi, `dig` sam odgrywa rolę resolvera rekurencyjnego: pyta serwery root, potem serwer TLD, potem serwer autorytatywny dla domeny — i pokazuje każdy krok osobno. Nieocenione, gdy podejrzewamy, że delegacja strefy jest błędna, a nie że coś jest nie tak z pojedynczym serwerem.

Przykład:

```
zielona.example.          172800  IN      NS      dc01.ad.zielona.example.
;; Received 68 bytes from 10.20.10.11#53(10.20.10.11) in 3 ms
app01.zielona.example.    3600    IN      A       10.20.10.41
;; Received 54 bytes from 10.20.10.11#53(dc01.ad.zielona.example) in 4 ms
```

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP ·**TLD** = Top Level Domain: najwyższy poziom nazwy domeny internetowej, część po ostatniej kropce, np. `.com` albo `.example` · **dig** = domain information groper: odpytywanie DNS

TYLKO ODCZYT

Sprawdzenie rekordów poczty: dig MX

```
dig MX NazwaDomeny +short
```

Pyta o rekordy MX (Mail Exchange) — które serwery przyjmują pocztę dla tej domeny i z jakim priorytetem (niższa liczba = wyższy priorytet).

Przykład:

```
10 poczta.zielona.example.
```

MX = Mail Exchanger: rekord DNS wskazujący serwer poczty domeny · **dig** = domain information groper: odpytywanie DNS

TYLKO ODCZYT

Sprawdzenie rekordów tekstowych: dig TXT

```
dig TXT NazwaDomeny +short
```

Rekordy TXT trzymają dowolny tekst — najczęściej wpisy SPF/DKIM/DMARC do weryfikacji poczty albo potwierdzenia własności domeny wymagane przez zewnętrzne usługi.

Przykład:

```
"v=spf1 mx ip4:203.0.113.10 -all"
```

TXT = typ rekordu DNS przechowującego dowolny tekst, wykorzystywany między innymi do publikowania reguł SPF, DKIM i DMARC · **SPF** = Sender Policy Framework: rekord DNS mówiący, które serwery mogą wysyłać pocztę domeny · **dig** = domain information groper: odpytywanie DNS

TYLKO ODCZYT

Sprawdzenie rekordów usług: dig SRV _ldap._tcp.NazwaDomeny

```
dig SRV _ldap._tcp.NazwaDomeny +short
```

Rekord SRV odpowiada na pytanie „który host i port obsługuje tę konkretną usługę w tej domenie" — bez niego klient musiałby znać adres kontrolera domeny na sztywno. Kluczowe przy Active Directory: kontrolery domeny, katalog LDAP i Kerberos są odnajdywane właśnie przez SRV, nie przez zwykłe rekordy A.

Przykład:

```
0 100 389 dc01.ad.zielona.example.
0 100 389 dc02.ad.zielona.example.
```

□□ **SRV** = rekord DNS wskazujący nie adres hosta, tylko usługę i port, pod którym jej szukać, np. kontroler domeny w sieci · **LDAP** = Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA) · **dig** = domain information groper: odpytywanie DNS

□□ **TYLKO ODCZYT**

Wykonanie zapytania odwrotnego: dig -x AdresIP

```
dig -x AdresIP +short
```

Odwrotność zwykłego zapytania: mamy adres IP, chcemy nazwę. Wymaga, żeby ktoś skonfigurował strefę odwrotną (`in-addr.arpa`) — jej brak jest bardzo częsty i wtedy komenda zwraca pustą odpowiedź, mimo że zapytanie „w przód” działa bez zarzutu.

Przykład:

```
app01.zielona.example.
```

□□ **IP** = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection) · **dig** = domain information groper: odpytywanie DNS

□□ **TYLKO ODCZYT**

Wysłanie zapytania do konkretnego serwera: dig @AdresIP NazwaDomeny

```
dig @AdresIP NazwaDomeny
```

Pomija `/etc/resolv.conf` i pyta wskazany serwer bezpośrednio. Niezastąpione przy porównywaniu dwóch serwerów DNS między sobą — np. gdy podejrzewamy, że rekord został zaktualizowany na jednym kontrolerze domeny, a na drugim jeszcze nie.

□□ **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP ·**dig** = domain information groper: odpytywanie DNS

□□ **TYLKO ODCZYT**

Wypisanie tylko sekcji odpowiedzi: dig +noall +answer

```
dig +noall +answer NazwaDomeny MX TXT
```

`+noall` wyłącza wszystkie domyślne sekcje, `+answer` włącza z powrotem tylko tę jedną. Wygodne przy pytaniu o kilka typów rekordów naraz — wynik nie ginie w powtarzającym się nagłówku.

Przykład:

```
app01.zielona.example.   3600    IN      A       10.20.10.41
```

dig = domain information groper: odpytywanie DNS · **MX** = Mail Exchanger: rekord DNS wskazujący serwer poczty domeny · **TXT** = typ rekordu DNS przechowującego dowolny tekst, wykorzystywany między innymi do publikowania reguł SPF, DKIM i DMARC

TYLKO ODCZYT

Wysłanie krótszego zapytania: host

```
host NazwaDomeny
```

Prostszy format wyjścia niż `dig`, jedna linia na rekord. Dobre do szybkiego sprawdzenia „na oko”, gorsze, gdy trzeba coś sparsować w skrypcie.

Przykład:

```
app01.zielona.example has address 10.20.10.41
```

TYLKO ODCZYT

Wybór typu rekordu: host -t

```
host -t MX NazwaDomeny
```

Odpowiednik `dig MX` w składni `host` — `-t` wybiera typ rekordu do odpytania.

Przykład:

```
zielona.example mail is handled by 10 poczta.zielona.example.
```

MX = Mail Exchanger: rekord DNS wskazujący serwer poczty domeny

TYLKO ODCZYT

Wysłanie zapytania przestarzałym narzędziem: nslookup

```
nslookup NazwaDomeny
```

Historycznie pierwsze narzędzie do zapytań DNS w Uniksie. Nadal działa i bywa jedynym dostępnym narzędziem na minimalnym obrazie kontenera, ale dokumentacja `dig`/`host` od lat opisuje `nslookup` jako przestarzałe — jego tryb interaktywny i format wyjścia bywają niespójne między implementacjami. Do nowych skryptów: `dig` albo `host`.

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **nslookup** = name server lookup: odpytywanie DNS

TYLKO ODCZYT

Sprawdzenie stanu resolvera systemowego: resolvectl status

```
resolvectl status
```

Pokazuje, którego serwera DNS faktycznie używa `systemd-resolved` na każdym interfejsie, oraz domenę wyszukiwania. To jest odpowiedź systemowa — inna niż to, co widzisz w `/etc/resolv.conf`, jeśli ten plik jest tylko dowiązaniem do stuba `systemd-resolved` (patrz *Jak to działa*).

Przykład:

```
Link 2 (ens192)
    Current Scopes: DNS
 DefaultRoute setting: yes
       Current DNS Server: 10.20.10.11
      DNS Servers: 10.20.10.11 10.20.10.12
       DNS Domain: ad.zielona.example
```

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP

TYLKO ODCZYT

Wysłanie zapytania przez resolver systemowy: resolvectl query

```
resolvectl query NazwaDomeny
```

W przeciwieństwie do `dig`, to zapytanie idzie dokładnie tą samą drogą, którą pójdzie każda zwykła aplikacja na tym systemie — przez `systemd-resolved`, z jego cache i jego kolejnością serwerów. Różnica wyniku między tym poleceniem a `dig` jest sygnałem, że problem leży w konfiguracji resolvera, nie w samym serwerze DNS.

Przykład:

```
app01.zielona.example: 10.20.10.41                    -- link: ens192
```

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP

TYLKO ODCZYT

Wyczyszczenie bufora resolvera: resolvectl flush-caches

```
sudo resolvectl flush-caches
```

Usuwa wszystkie zapamiętane odpowiedzi z lokalnego cache'u `systemd-resolved`. Pierwsza rzecz do zrobienia, gdy rekord DNS został właśnie zmieniony na serwerze, a lokalna maszyna uparcie zwraca starą wartość.

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP

TYLKO ODCZYT

Sprawdzenie dokładnie tego, co zobaczy aplikacja: getent hosts

```
getent hosts NazwaDomeny
```

To jedyne polecenie z tej Fiszki, które przechodzi przez **cały** mechanizm rozwiązywania nazw systemu — łącznie z `/etc/hosts` i kolejnością źródeł z `/etc/nsswitch.conf`. Jeśli chcemy wiedzieć, na jaki adres faktycznie połączy się dowolna aplikacja na tej maszynie, to jest właściwe pytanie — nie `dig`.

Przykład:

```
10.20.10.41     app01.zielona.example
```

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

TYLKO ODCZYT

Sprawdzenie skonfigurowanych serwerów DNS: cat /etc/resolv.conf

```
cat /etc/resolv.conf
```

Lista serwerów DNS i domen wyszukiwania używanych przez klasyczny resolver systemowy (bibliotekę `glibc`). Na systemach z `systemd-resolved` ten plik bywa dowiązaniem symbolicznym do stuba `127.0.0.53` — patrz *Jak to działa*.

Przykład:

```
# Generated by NetworkManager
nameserver 127.0.0.53
options edns0 trust-ad
search ad.zielona.example
```

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **glibc** = GNU C Library: podstawowa biblioteka systemowa Linuksa, od której wersji zależy, czy dany program (na przykład serwer P4) w ogóle się uruchomi · **cat** = concatenate: sklej i wypisz pliki

TYLKO ODCZYT

Sprawdzenie kolejności źródeł nazw: cat /etc/nsswitch.conf

```
grep ^hosts /etc/nsswitch.conf
```

Definiuje, w jakiej kolejności system szuka odpowiedzi na pytanie „jaki adres ma ta nazwa” — plik, DNS, czy jeszcze coś innego (np. `sss` przy integracji z Active Directory). Pełne omówienie w *Jak to działa*.

Przykład:

```
hosts: files sss dns myhostname
```

DNS = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **grep** = global regular expression print: szukaj wzorca w tekście

TYLKO ODCZYT

Sprawdzenie stanu usługi systemd-resolved

```
systemctl status systemd-resolved --no-pager | head -5
```

Na RHEL 9/10 i nowszym Ubuntu `NetworkManager` nie rozwiązuje nazw samodzielnie — przekazuje adresy serwerów DNS otrzymane przez DHCP do `systemd-resolved`, a ten prowadzi cache i wybiera, którego serwera użyć na którym interfejsie. SLES i starsze wydania Ubuntu bywają skonfigurowane inaczej — `resolvectl status` zawsze pokaże, co faktycznie jest aktywne, niezależnie od tego, która kombinacja usług za tym stoi.

Przykład:

```
● systemd-resolved.service - Network Name Resolution
     Active: active (running)
```

RHEL = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **DHCP** = Dynamic Host Configuration Protocol: usługa, która sama rozdaje komputerom adresy IP · **SLES** = SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL · **systemctl** = system control: sterowanie usługami systemd

TYLKO ODCZYT

Wszystkie flagi i komendy

Komenda	Co robi
`dig nazwa`	pełne zapytanie bezpośrednio do serwera DNS
`dig +short`	sam wynik, bez nagłówków
`dig +trace`	cała droga od serwerów root do autorytatywnego
`dig MX` / `TXT` / `SRV`	konkretny typ rekordu
`dig -x IP`	zapytanie odwrotne (IP → nazwa)
`dig @serwer nazwa`	zapytanie do konkretnego, wskazanego serwera
`dig +noall +answer`	tylko sekcja odpowiedzi
`host` / `host -t`	krótszy format, z wyborem typu rekordu
`nslookup`	działa, ale przestarzałe — nie do nowych skryptów
`resolvectl status`	który serwer DNS jest faktycznie używany
`resolvectl query`	zapytanie tą samą drogą co aplikacje
`resolvectl flush-caches`	czyszczenie lokalnego cache'u
`getent hosts`	to, co naprawdę zobaczy aplikacja (z `/etc/hosts`)
`cat /etc/resolv.conf`	skonfigurowane serwery DNS
`cat /etc/nsswitch.conf`	kolejność źródeł rozwiązywania nazw

Uważaj

- `dig` **omija** `/etc/hosts` i `nsswitch.conf` — wynik `dig` i wynik `getent hosts` mogą się różnić, i oba mogą być „poprawne” na swój sposób.
- `dig +trace` generuje wiele zapytań do serwerów root — nie uruchamiaj tego w pętli ani często w automatyzacji.
- `resolvectl` istnieje tylko tam, gdzie działa `systemd-resolved` — na systemie bez tej usługi (część konfiguracji SLES, kontenery) polecenie nie istnieje.

□□ **MX** = Mail Exchanger: rekord DNS wskazujący serwer poczty domeny · **TXT** = typ rekordu DNS przechowującego dowolny tekst, wykorzystywany między innymi do publikowania reguł SPF, DKIM i DMARC · **IP** = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection) · **DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **SLES** = SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

Czwartek, 10:00. Aplikacja na app01 nie może połączyć się z bazą na pg01 po nazwie, choć adres IP działa bez problemu. Tomek sprawdza dig i zauważa, że wpis w /etc/hosts nadpisuje starą, już nieaktualną nazwę serwera, więc resolver w ogóle nie pyta o to DNS-u kontrolera domeny.

Po co to istnieje

Pojęcie: rozwiązywanie nazw. Rozwiązywanie nazw to zamiana nazwy czytelnej dla człowieka (`app01.zielona.example`) na adres IP, którego faktycznie potrzebuje warstwa sieciowa, żeby nawiązać połączenie. System nie ma jednego, sztywno zaszytego sposobu na to zadanie — ma **kolejność źródeł**, którą sam administrator konfiguruje w pliku `/etc/nsswitch.conf`: najpierw sprawdź lokalny plik `/etc/hosts`, potem zapytaj usługę integracji z domeną, potem dopiero zapytaj serwer DNS, i tak dalej, w kolejności zapisanej w tym pliku. To rozróżnienie — narzędzie, które pyta serwer DNS bezpośrednio, kontra mechanizm, którego naprawdę używa aplikacja — jest źródłem większości nieporozumień opisanych na tej stronie.

Administrator sięga po `dig`, `host` i `resolvectl`, kiedy coś w komunikacji między nazwą a adresem nie działa: aplikacja łączy się z niewłaściwym serwerem, przeglądarka zgłasza błąd rozwiązywania nazwy, albo zmiana rekordu DNS „nie działa”, mimo że administrator DNS przysięga, że wpis jest poprawny od godziny.

Analogia

Wyobraź sobie, że szukasz numeru telefonu do znajomego na dwa sposoby: **zaglądasz do własnej papierowej książki adresowej** albo **dzwonisz do biura numerów**.

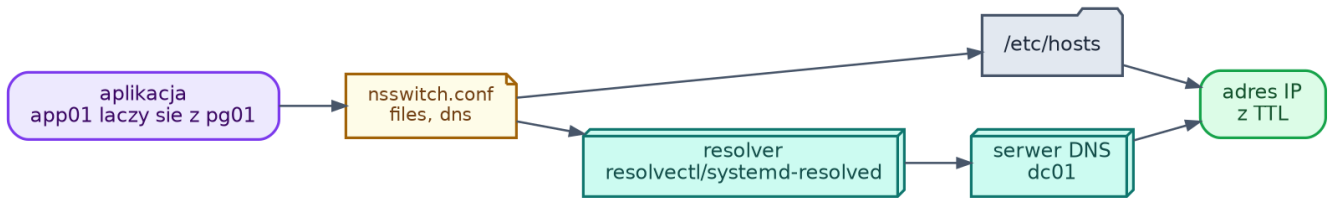
Twoja papierowa książka adresowa to `/etc/hosts` — pierwsze miejsce, do którego zaglądasz, bo jest pod ręką. Jeśli wpis tam jest, sprawa jest zamknięta, nawet jeśli jest już nieaktualny — znajomy dawno zmienił numer, a Ty nadal dzwonisz na stary, bo książka mówiła, że to ten.

Biuro numerów to serwer DNS. Zapytasz je wprost, dostaniesz aktualną odpowiedź prosto ze źródła — to robi `dig`, pytając serwer bezpośrednio, z pominięciem własnej książki adresowej. Ale to nie jest droga, którą idzie zwykły telefon: najpierw sprawdza zapisane kontakty (`/etc/hosts`), i dopiero gdy nic tam nie znajdzie, sam dzwoni do biura numerów w Twoim imieniu (zapytanie DNS przez `nsswitch.conf`). `dig` to Ty, dzwoniący do biura numerów osobiście — dlatego może pokazać coś innego, niż zobaczy Twój telefon.

Jak to działa

Rozwiń: jak to działa pod spodem

> **Pojęcie: kolejność źródeł nazw (nsswitch).** Linux nie ma jednego wbudowanego mechanizmu rozwiązywania nazw — ma bibliotekę `glibc` (Name Service Switch), która przy każdym zapytaniu o nazwę czyta plik `/etc/nsswitch.conf` i sprawdza źródła w kolejności tam zapisanej. Typowy wpis wygląda tak:



Kolejność w nsswitch.conf decyduje, czy odpowiedź przyjdzie z lokalnego pliku hosts, czy dopiero z serwera DNS. (kliknięcie otwiera pełny rozmiar)

```
hosts: files sss dns myhostname
```

Cztery źródła sprawdzane po kolei: `files` (`/etc/hosts`), `sss` (SSSD — integracja z Active Directory/FreeIPA, jeśli system jest do takiej domeny dołączony), `dns` (zwykłe zapytanie do serwera DNS) i `myhostname` (nazwa samej maszyny, na wypadek gdyby ktoś pytał o siebie). Sprawdzanie kończy się na pierwszym źródle, które daje jednoznaczną odpowiedź — wpis w `/etc/hosts` wygrywa z prawdziwym rekordem DNS, nawet jeśli jest nieaktualny.

> **Pojęcie: rekord DNS i typy rekordów.** Rekord DNS to pojedynczy wpis w bazie danych serwera nazw, wiążący nazwę z jakąś informacją — nie zawsze z adresem IP. Typ rekordu określa, jakiego rodzaju informację ten wpis niesie: `A` wiąże nazwę z adresem IPv4, `AAAA` z adresem IPv6, `MX` wskazuje serwer poczty dla domeny, `TXT` niesie dowolny tekst (najczęściej dane weryfikacyjne SPF/DKIM), `SRV` wskazuje host i port konkretnej usługi, `PTR` jest odwrotnością `A` — wiąże adres IP z nazwą. Fiszka wyżej pokazuje zapytania o pięć z tych typów, bo to te, z którymi administrator styka się najczęściej.

> **Pojęcie: TTL i buforowanie.** Każdy rekord DNS ma przypisany TTL (Time To Live) — czas w sekundach, przez jaki odpowiedź wolno zapamiętać (zbuforować) bez ponownego pytania serwera autorytatywnego. Serwer DNS, resolver systemowy i czasem sama aplikacja trzymają taką kopię w pamięci. To wyjaśnia klasyczną sytuację: administrator zmienia rekord na serwerze DNS, a klient przez kolejne minuty czy godziny (w zależności od TTL) nadal łączy się pod starym adresem — nie dlatego, że coś jest zepsute, tylko dlatego, że wciąż używa zapamiętanej, jeszcze ważnej według TTL odpowiedzi.

> **Pojęcie: resolver lokalny.** Resolver lokalny to proces działający na samej maszynie, przez który przechodzą zapytania DNS aplikacji, zanim (albo zamiast) trafią na zewnętrzny serwer. Na RHEL 9/10 tę rolę pełni `systemd-resolved`: nasłuchuje na `127.0.0.53`, prowadzi własny cache i decyduje, którego z faktycznie skonfigurowanych serwerów DNS użyć dla którego interfejsu. Właśnie dlatego `/etc/resolv.conf` na takim systemie często zawiera tylko `nameserver 127.0.0.53` — to nie jest prawdziwy serwer DNS, to stub wskazujący na resolver lokalny, a prawdziwe serwery widać dopiero przez `resolvectl status`.

Dlaczego „`dig` działa, a aplikacja nie”. To najczęstsza niespójność opisana na tej stronie i wynika wprost z dwóch poprzednich pojęć. `dig` **pomija** zarówno `/etc/hosts`, jak i całą logikę `nsswitch.conf` — pyta serwer wzięty z `/etc/resolv.conf` (albo podany jawnie przez `@serwer`) bezpośrednio. Zwykła aplikacja — przeglądarka, `curl` bez specjalnych flag, klient bazy danych

— idzie przez `glibc` i pełną kolejność z `nsswitch.conf`. Jeśli w `/etc/hosts` jest przestarzały wpis, jeśli `systemd-resolved` ma w cache'u starą odpowiedź, albo SSSD zwraca coś innego niż zwykły DNS — `dig` i aplikacja dostaną **różne** odpowiedzi na dokładnie to samo pytanie. Sprawdzenie tego, co naprawdę zobaczy aplikacja, to zadanie dla `getent hosts`, nie dla `dig`.

Rekordy SRV i Active Directory. Klient dołączający do domeny Active Directory (czy to przez `realmd`, czy ręcznie przez `sssd`) nie ma zaszytego na sztywno adresu kontrolera domeny — znajduje go, pytając DNS o konkretne rekordy SRV, m.in. `_ldap._tcp.<domena>` (katalog LDAP) i `_kerberos._tcp.<domena>` (uwierzytelnianie Kerberos). Jeśli te rekordy nie istnieją albo wskazują nieaktywny kontroler, dołączenie do domeny kończy się niejasnym błędem, mimo że zwykłe `dig nazwa-kontrolera` działa bez zarzutu. Pełna procedura dołączania do domeny i wymagania dotyczące DNS: *Dodanie systemu Linux do domeny AD poprzez RealmD*.

Krok po kroku

1. Sprawdź, co faktycznie jest skonfigurowane jako serwer DNS.

```
resolvectl status | grep -A2 'Current DNS'
```

```
Current DNS Server: 10.20.10.11
DNS Servers: 10.20.10.11 10.20.10.12
```

2. Zapytaj bezpośrednio ten serwer o rekord, z którym jest problem.

```
dig @10.20.10.11 app01.zielona.example +noall +answer
```

```
app01.zielona.example. 300   IN      A       10.20.10.41
```

Serwer zwraca poprawny, aktualny adres.

3. Sprawdź, co faktycznie zobaczy aplikacja na tej maszynie.

```
getent hosts app01.zielona.example
```

```
10.20.10.141    app01.zielona.example
```

Inny adres niż w kroku 2 — to jest właściwy moment, w którym problem przestaje być problemem DNS.

4. Znajdź, które źródło z `nsswitch.conf` wygrywa.

```
grep ^hosts /etc/nsswitch.conf
grep -i aplikacja /etc/hosts
```

```
hosts: files sss dns myhostname
10.20.10.141   app01.zielona.example   # dopisane ręcznie w marcu
```

Wpis w `/etc/hosts`, zapomniany od miesięcy, wygrywa z prawdziwym rekordem DNS, bo `files` jest sprawdzane przed `dns`.

5. Napraw i potwierdź.

```
sudo sed -i '/app01.zielona.example/d' /etc/hosts
getent hosts app01.zielona.example
```

```
10.20.10.41     app01.zielona.example
```

Częste błędy

„`dig` pokazuje dobry adres, aplikacja się łączy ze złym”.

```
dig app01.zielona.example +short
getent hosts app01.zielona.example
```

```
10.20.10.41
10.20.10.141    app01.zielona.example
```

Dwie różne odpowiedzi na to samo pytanie — bo to w gruncie rzeczy dwa różne pytania. `dig` odpytał serwer DNS bezpośrednio; `getent hosts` przeszedł przez pełną kolejność z `nsswitch.conf`, w tym wypadku trafiając na wpis w `/etc/hosts`. Naprawa: znaleźć i usunąć (albo poprawić) przestarzały wpis w `/etc/hosts`, nie winić serwera DNS.

Zapytanie odwrotne zwraca pustkę, mimo że zapytanie w przód działa.

```
dig -x 10.20.10.41 +short
```

```
(brak wyniku)
```

Rekord `A` istnieje, ale nikt nie skonfigurował odpowiadającej mu strefy odwrotnej (`in-addr.arpa`) z rekordem `PTR`. To dwa niezależne wpisy w bazie serwera DNS — istnienie jednego nie gwarantuje istnienia drugiego. Naprawa: administrator strefy DNS musi dodać rekord PTR osobno; sam klient

nic tu nie naprawi.

Zmieniony rekord „nie działa” od razu po edycji.

```
dig app01.zielona.example +short
```

```
10.20.10.41
```

Administrator dopiero co zmienił ten rekord na `10.20.10.55` na serwerze DNS, a `dig` uparcie zwraca starą wartość. Przyczyna: TTL poprzedniej odpowiedzi jeszcze nie wygasł, więc resolver (albo sam serwer pośredniczący) wciąż serwuje zapamiętaną wartość. Naprawa: poczekać do wygaśnięcia TTL albo, na maszynie klienckiej, wymusić czyszczenie cache'u: `sudo resolvectl flush-caches`.

`nslookup` w skrypcie parsowany przez `grep`/`awk` i rozjeżdżający się format.

```
nslookup app01.zielona.example | grep Address | tail -1 | awk '{print $2}'
```

Działa na jednym systemie i psuje się na innym, bo dokładny format wyjścia `nslookup` różni się między implementacjami (a bywa też zlokalizowany na inny język w zależności od ustawień systemu). Naprawa: `dig +short app01.zielona.example` daje deterministyczny, przeznaczony do parsowania format, bez zgadywania, na której linii jest odpowiedź.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz lokalnie. Zapytania o publiczne domeny (np. `example.com`) wymagają działającego łącza do internetu; edycja `/etc/hosts` wymaga uprawnień administratora i jest w pełni odwracalna.

Ćwiczenie 1 — sprzeczność między `/etc/hosts` a DNS na własnej maszynie

```
echo "127.0.0.10 test-lokalny.invalid" | sudo tee -a /etc/hosts
dig test-lokalny.invalid +short
getent hosts test-lokalny.invalid
```

Dlaczego pierwsze polecenie nic nie zwraca, a drugie zwraca adres, mimo że pytają o tę samą nazwę?

Odpowiedź

`dig` pyta serwer DNS wskazany w `/etc/resolv.conf` — a ten nigdy nie słyszał o `test-lokalny.invalid`, bo to nazwa istniejąca tylko lokalnie, w pliku `/etc/hosts`. `getent hosts` przechodzi przez pełną kolejność z `nsswitch.conf`, w której `files` jest sprawdzane jako pierwsze — i tam ta nazwa istnieje. Posprzątaj:

```
sudo sed -i '/test-lokalny.invalid/d' /etc/hosts
```

Ćwiczenie 2 — TTL i buforowanie na żywo

```
dig example.com +noall +answer | awk '{print $2}'
```

Uruchom tę komendę dwa razy w odstępie kilku sekund, a potem poczekaj tyle sekund, ile pokazywał wynik, i uruchom ponownie. Co się zmienia w liczbie, i dlaczego przy trzecim uruchomieniu wynik jest znów wysoki?

Odpowiedź

Druga liczba (TTL) maleje przy kolejnych zapytaniach w krótkim odstępie — to dowód, że odpowiedź jest serwowana z pamięci podręcznej resolvera, nie pobierana od nowa z serwera autorytatywnego za każdym razem. Po wygaśnięciu TTL do zera resolver musi zapytać serwer ponownie, dostaje świeżą odpowiedź z nowym, pełnym TTL, i licznik zaczyna się od nowa.

Pytania kontrolne

1. Dlaczego `dig nazwa` i `getent hosts nazwa` mogą zwrócić dwa różne adresy dla tej samej nazwy?

Odpowiedź

Bo `dig` pyta serwer DNS bezpośrednio, pomijając `/etc/hosts` i `nsswitch.conf`, a `getent hosts` przechodzi przez pełną, skonfigurowaną kolejność źródeł — w której `/etc/hosts` (albo integracja z domeną przez SSSD) może wygrać z prawdziwym rekordem DNS.

2. Które polecenie pokaże dokładnie to, na jaki adres połączy się przeglądarka na tej maszynie: `dig`, `host`, czy `getent hosts`?

Odpowiedź

`getent hosts` — to jedyne z trzech, które przechodzi przez ten sam mechanizm rozwiązywania nazw (bibliotekę `glibc` i `nsswitch.conf`), z którego korzysta każda zwykła aplikacja, w tym przeglądarka. `dig` i `host` pytają serwer DNS bezpośrednio, z pominięciem tej ścieżki.

3. Administrator dołącza serwer do domeny Active Directory i dostaje błąd, mimo że `dig dc01.ad.zielona.example` zwraca poprawny adres. Jakiego typu rekordu prawdopodobnie brakuje?

Odpowiedź

Rekordu SRV — np. `_ldap._tcp.ad.zielona.example` albo `_kerberos._tcp.ad.zielona.example`. Proces dołączania do domeny nie znajduje kontrolera po nazwie wpisanej na sztywno; odpytuje DNS o te konkretne rekordy usługowe, żeby ustalić, który host i port obsługuje LDAP czy Kerberos. Zwykły rekord `A` działającego kontrolera nie wystarcza, jeśli brakuje odpowiadających mu rekordów SRV.

4. Zmieniono adres serwera pocztowego w rekordzie DNS, a `dig` wciąż zwraca stary adres pięć minut później. Czy to błąd w konfiguracji serwera DNS?

Odpowiedź

Niekoniecznie. To typowy efekt TTL — poprzednia odpowiedź mogła mieć TTL większy niż pięć minut i wciąż jest ważna według resolvera, który ją zapamiętał. Dopiero po wygaśnięciu TTL resolver zapyta serwer ponownie. Do natychmiastowego sprawdzenia nowej wartości: zapytanie bezpośrednio do serwera autorytatywnego (`dig @serwer ...`) albo wymuszone czyszczenie lokalnego cache'u (`resolvectl flush-caches`).

Powiązane: Dodanie systemu Linux do domeny AD poprzez RealmD · Diagnostyka: ping, traceroute, mtr, tcpdump, curl · Warstwa pod firewalld: nftables i iptables **Źródła:** `man 1 dig` · `man 1 host` · `man 1 resolvectl` · `man 5 nsswitch.conf` · `man 5 resolv.conf`

Revision #16
Created 2026-09-17 21:43:23 UTC by Tomasz Konieczny
Updated 2026-10-03 19:09:06 UTC by Tomasz Konieczny