firewalld: strefy, porty, usługi, rich rules

◫◫ Poziom: średnio

◫◫ Dotyczy: RHEL/Rocky 9–10 i SLES (firewalld domyślnie), Ubuntu zwykle przez ufw, firewalld opcjonalny

▯ Czas: ~22 min

`firewalld` to zapora, która trzyma dwie osobne kopie reguł — bieżącą i trwałą — i to jest źródło niemal wszystkich pomyłek na tej stronie. Reguła bez `--permanent` znika po `--reload`. Reguła z `--permanent` bez `--reload` nie działa wcale.

◫◫ Skróty użyte na tej stronie (13)

Skrót	Co znaczy
FTP	File Transfer Protocol: stary protokół przesyłania plików, bez szyfrowania
HTTP	Hypertext Transfer Protocol: protokół stron WWW
ICMP	Internet Control Message Protocol: protokół diagnostyczny sieci, którym działa między innymi ping
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
NAT	Network Address Translation: podmiana adresów IP na granicy sieci
NTP	Network Time Protocol: synchronizacja zegara po sieci
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
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia
UDP	User Datagram Protocol: szybki protokół sieciowy bez potwierdzeń
VRRP	Virtual Router Redundancy Protocol: protokół, w którym kilka hostów uzgadnia, który z nich trzyma wspólny adres IP (opisany w RFC 5798)
XML	eXtensible Markup Language: format tekstowy do zapisu ustrukturyzowanych danych w znacznikach, tutaj używany przez PowerShell do zapisu zrzutu obiektów poleceniem Export-Clixml

Fiszka

□□ TYLKO ODCZYT

Sprawdzenie, czy zapora działa: firewall-cmd --state

```
firewall-cmd --state
```

Najprostsze możliwe pytanie, zanim zaczniemy cokolwiek diagnozować głębiej.

Przykład:

```
running
```

□□ **firewall-cmd** = sterowanie zaporą firewalld

□□ TYLKO ODCZYT

Sprawdzenie wszystkich reguł bieżącej strefy: firewall-cmd --list-all

```
sudo firewall-cmd --list-all
```

Pokazuje **bieżącą, działającą w tej chwili** konfigurację domyślnej strefy: interfejsy, usługi, porty, rich rules, maskaradę. To jest stan runtime — nie pokazuje automatycznie tego, co jest tylko `--permanent` i czeka na `--reload`.

Przykład:

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens192
  sources:
  services: ssh dhcpv6-client
  ports: 443/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
```

```
  source-ports:
  icmp-blocks:
  rich rules:
```

firewall-cmd = sterowanie zaporą firewalld

TYLKO ODCZYT

Sprawdzenie wszystkich reguł we wszystkich strefach: firewall-cmd --list-all-zones

```
sudo firewall-cmd --list-all-zones
```

To samo co wyżej, ale dla każdej zdefiniowanej strefy z osobna — przydatne, gdy podejrzewamy, że reguła trafiła do niewłaściwej strefy (patrz *Częste błędy*).

Przykład:

```
public (active)
  target: default
  interfaces: ens192
  services: ssh dhcpv6-client
  ports: 443/tcp
...
drop
  target: DROP
  interfaces:
  services:
  ports:
...
```

firewall-cmd = sterowanie zaporą firewalld

TYLKO ODCZYT

Sprawdzenie, która strefa obsługuje dany interfejs: firewall-cmd --get-active-zones

```
firewall-cmd --get-active-zones
```

> **Pojęcie: strefa firewalld (zone).** Nazwany zestaw reguł przypisany do jednego lub więcej interfejsów sieciowych — `public` (nieufna sieć zewnętrzna), `internal`/`trusted` (sieć lokalna, mniej restrykcyjna), `drop` (odrzuca wszystko bez odpowiedzi). Zamiast pisać osobne reguły dla każdego interfejsu, przypisujemy interfejs do strefy o gotowym, sensownym profilu restrykcyjności. To polecenie odpowiada wprost, który interfejs jest w której strefie — pierwsze, co warto sprawdzić, zanim zaczniemy dodawać reguły „do strefy X", jeśli nie mamy pewności, że to w ogóle strefa obsługująca nasz interfejs.

Przykład:

```
public
  interfaces: ens192
```

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Sprawdzenie i zmiana strefy domyślnej: firewall-cmd --get-default-zone / --set-default-zone

```
firewall-cmd --get-default-zone
sudo firewall-cmd --set-default-zone=NazwaZony
```

Strefa domyślna to ta, do której trafia interfejs, jeśli nic innego (NetworkManager, profil połączenia) nie powie inaczej. Zmiana strefy domyślnej **nie przenosi** już podłączonych interfejsów — działa tylko na przyszłość, chyba że dany interfejs jawnie trzyma się strefy domyślnej.

Przykład:

```
public
```

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie portu trwale: firewall-cmd --add-port=NumerPortu/tcp --permanent

```
sudo firewall-cmd --add-port=NumerPortu/tcp --permanent
```

Zapisuje regułę do pliku konfiguracji trwałej — **nie stosuje jej do ruchu na żywo**. Do tego potrzebny jest `--reload` (albo `--runtime-to-permanent` w drugą stronę, patrz niżej). To jest dokładnie ta pułapka opisana we wstępie strony.

Przykład:

```
success
```

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie usługi: firewall-cmd --add-service=NazwaService --permanent

```
sudo firewall-cmd --add-service=NazwaService --permanent
sudo firewall-cmd --reload
```

> **Pojęcie: usługa firewalld jako nazwany zestaw portów.** Zamiast pamiętać, że HTTP to port 80/tcp, a NTP to 123/udp, `firewalld` ma wbudowane definicje usług — pliki XML mapujące przyjazną nazwę na konkretny port i protokół (czasem kilka naraz, jak przy FTP z jego dynamicznym zakresem portów). `--add-service=http` jest równoważne dodaniu portu `80/tcp`, ale czytelniejsze w audycie reguł i odporne na pomyłkę w numerze.

HTTP = Hypertext Transfer Protocol: protokół stron WWW · **NTP** = Network Time Protocol: synchronizacja zegara po sieci · **XML** = eXtensible Markup Language: format tekstowy do zapisu ustrukturyzowanych danych w znacznikach, tutaj używany przez PowerShell do zapisu zrzutu obiektów poleceniem Export-Clixml · **FTP** = File Transfer Protocol: stary protokół przesyłania plików, bez szyfrowania · **firewall-cmd** = sterowanie zaporą firewalld

OSTROŻNIE · NIEODWRACALNE

Usunięcie portu: firewall-cmd --remove-port=NumerPortu/tcp --permanent

```
sudo firewall-cmd --remove-port=NumerPortu/tcp --permanent
sudo firewall-cmd --reload
```

Symetryczne do dodawania — i tak samo wymaga `--reload`, żeby usunięcie faktycznie zamknęło port w ruchu na żywo, a nie tylko w pliku konfiguracji.

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie portu do testu, bez --permanent: firewall-cmd --add-port=NumerPortu/tcp

```
sudo firewall-cmd --add-port=NumerPortu/tcp
```

Bez `--permanent` reguła działa **od razu**, na żywym ruchu, bez żadnego `--reload` — ale zniknie przy najbliższym `--reload` albo restarcie usługi `firewalld`/maszyny. To jest właściwy sposób testowania: sprawdzamy, czy port faktycznie działa, zanim zdecydujemy się utrwalić regułę.

Przykład:

```
success
```

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie reguły dla protokołu bez dedykowanej usługi: firewall-cmd --add-rich-rule

```
sudo firewall-cmd --add-rich-rule='rule protocol value="vrrp" accept' --permanent
sudo firewall-cmd --reload
```

VRRP (Virtual Router Redundancy Protocol, protokół IP numer 112) nie jest ani portem TCP /UDP, ani zdefiniowaną usługą `firewalld` — używają go klastry wysokiej dostępności (np. `keepalived`) do wymiany komunikatów o tym, który węzeł aktualnie trzyma adres wirtualny. Rich rule z `protocol value="vrrp"` wpuszcza ten konkretny protokół IP, czego nie da się wyrazić przez zwykłe `--add-port` (bo VRRP nie ma numeru portu — działa bezpośrednio na warstwie IP).

Przykład:

```
success
```

VRRP = Virtual Router Redundancy Protocol: protokół, w którym kilka hostów uzgadnia, który z nich trzyma wspólny adres IP (opisany w RFC 5798) · **IP** = Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection) · **TCP** = Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia · **firewall-cmd** = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie dostępu do usługi tylko z jednego źródła: firewall-cmd --add-rich-rule

```
sudo firewall-cmd --add-rich-rule='rule family="ipv4" source address="AdresIP/32"
service name="ssh" accept' --permanent
sudo firewall-cmd --reload
```

Pojęcie: konfiguracja bieżąca kontra trwała. `firewalld` utrzymuje dwie osobne kopie zestawu reguł: **runtime** — to, co faktycznie filtruje ruch w tej chwili (żyje w pamięci, budowane na starcie usługi z konfiguracji trwałej) — i **permanent** — pliki w `/etc/firewalld/`, wczytywane na starcie usługi albo po `--reload`. Rich rule pozwala łączyć więcej warunków niż zwykły port czy usługa: tu — konkretny adres źródłowy **razem z** konkretną usługą, czyli „SSH wolno tylko z tego jednego hosta administracyjnego", zamiast otwierać SSH dla wszystkich.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **firewall-cmd** = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie logowania i odrzucenia z komunikatem: firewall-cmd --add-rich-rule

```
sudo firewall-cmd --add-rich-rule='rule family="ipv4" source address="AdresIP/24" log
prefix="blocked-subnet: " level="warning" reject' --permanent
sudo firewall-cmd --reload
```

`log` zapisuje dopasowany ruch do dziennika jądra (widoczny przez `journalctl -k` albo `dmesg`) z podanym prefiksem, ułatwiającym późniejsze `grep`. `reject` różni się od `drop` tym, że odsyła nadawcy jawną odpowiedź odrzucenia (np. ICMP port unreachable) zamiast po prostu wyciszać pakiet bez żadnej reakcji — druga strona wie od razu, że połączenie zostało zablokowane, zamiast czekać na przekroczenie limitu czasu.

ICMP = Internet Control Message Protocol: protokół diagnostyczny sieci, którym działa między innymi ping · **firewall-cmd** = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie reguły, która sama się cofnie: firewall-cmd --add-rich-rule ... --timeout=60s

```
sudo firewall-cmd --add-rich-rule='rule family="ipv4" source address="0.0.0.0/0" service name="ssh" reject' --timeout=60s
```

Zabezpieczenie przed zamknięciem sobie dostępu. `--timeout` działa tylko na regułę runtime (nie łączy się z `--permanent`) i sam ją usuwa po podanym czasie — jeśli testujemy coś ryzykownego, np. regułę mogącą zablokować nasze własne połączenie SSH, `--timeout` gwarantuje, że nawet przy najgorszym scenariuszu dostęp wróci sam, bez konieczności fizycznego dostępu do konsoli.

Przykład:

```
success
```

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **firewall-cmd** = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Przypisanie źródła do strefy: firewall-cmd --add-source=AdresIP/24 --zone=NazwaZony --permanent

```
sudo firewall-cmd --add-source=AdresIP/24 --zone=NazwaZony --permanent
sudo firewall-cmd --reload
```

Zamiast przypisywać całą strefę do interfejsu, można przypisać ją do konkretnej podsieci źródłowej — ruch z `AdresIP/24` trafia do reguł strefy `NazwaZony` niezależnie od tego, którym interfejsem faktycznie wszedł.

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Przeniesienie interfejsu do innej strefy: firewall-cmd --change-interface=NazwaInterfejsu --zone=NazwaZony

```
sudo firewall-cmd --change-interface=NazwaInterfejsu --zone=NazwaZony --permanent
sudo firewall-cmd --reload
```

Zmienia przypisanie interfejsu do strefy — inne niż to, które nadał domyślnie NetworkManager przy podłączeniu (zwykle strefa domyślna albo, dla profili z ustawionym `connection.zone`, strefa wskazana w profilu `nmcli`).

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Włączenie maskarady (NAT źródłowy): firewall-cmd --add-masquerade --permanent

```
sudo firewall-cmd --zone=NazwaZony --add-masquerade --permanent
sudo firewall-cmd --reload
```

Włącza translację adresów źródłowych (source NAT) dla ruchu wychodzącego przez tę strefę — konieczne, gdy maszyna ma pełnić rolę routera/bramy dla innych hostów, np. dla kontenerów albo maszyn w sieci wewnętrznej korzystających z tego serwera jako wyjścia do internetu.

NAT = Network Address Translation: podmiana adresów IP na granicy sieci · **firewall-cmd** = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Dodanie przekierowania portu: firewall-cmd --add-forward-port

```
sudo firewall-cmd --zone=NazwaZony --add-forward-
port=port=NumerPortu:proto=tcp:toport=NumerPortu:toaddr=AdresIP --permanent
sudo firewall-cmd --reload
```

> (gdzie pierwszy `NumerPortu` to port, na który przychodzi ruch na tej maszynie, a drugi — port na hoście docelowym `AdresIP`)

Ruch przychodzący na port 8080 tej maszyny jest przekierowywany do portu 80 na innym hoście (`10.20.10.41`) — klasyczne użycie: brama/serwer wystawiony publicznie kieruje ruch do usługi ukrytej w sieci wewnętrznej.

firewall-cmd = sterowanie zaporą firewalld

ZMIENIA SYSTEM

Zastosowanie trwałej konfiguracji: firewall-cmd --reload

```
sudo firewall-cmd --reload
```

Wczytuje konfigurację `--permanent` z `/etc/firewalld/` i nadpisuje nią bieżący stan runtime. **Każda reguła dodana bez `--permanent` i bez uprzedniego zapisania ginie w tym momencie** — to jest moment, w którym testowe, tymczasowe reguły znikają.

Przykład:

```
success
```

firewall-cmd = sterowanie zaporą firewalld

TYLKO ODCZYT

Zapisanie tego, co już przetestowaliśmy: firewall-cmd --runtime-to-permanent

```
sudo firewall-cmd --runtime-to-permanent
```

Odwrotny kierunek: bierze **cały aktualny stan runtime** (łącznie z regułami dodanymi bez `--permanent`, które akurat teraz działają) i zapisuje go jako nową konfigurację trwałą. Wygodne wyjście z sytuacji „przetestowałem na żywo kilka reguł, działają, nie chcę teraz ręcznie przepisywać każdej z osobna z `--permanent`".

Przykład:

```
success
```

firewall-cmd = sterowanie zaporą firewalld

TYLKO ODCZYT

Sprawdzenie usług w strefie i usług w ogóle dostępnych: firewall-cmd --list-services / --get-services

```
firewall-cmd --list-services
firewall-cmd --get-services | tr ' ' '\n' | head -10
```

`--list-services` pokazuje usługi włączone w bieżącej (albo wskazanej `--zone=`) strefie. `--get-services` wypisuje wszystkie nazwy usług, jakie `firewalld` w ogóle zna — zdefiniowane w plikach XML dostarczanych z pakietem, niezależnie od tego, czy są gdziekolwiek włączone.

Przykład:

```
ssh dhcpv6-client http
```

Przykład:

```
RH-Satellite-6
amanda-client
amanda-k5-client
bacula
bacula-client
bgp
bitcoin
```

XML = eXtensible Markup Language: format tekstowy do zapisu ustrukturyzowanych danych w znacznikach, tutaj używany przez PowerShell do zapisu zrzutu obiektów poleceniem Export-Clixml · **firewall-cmd** = sterowanie zaporą firewalld

TYLKO ODCZYT

Sprawdzenie szczegółów jednej usługi: firewall-cmd --info-service=NazwaService

```
firewall-cmd --info-service=NazwaService
```

Dokładnie jakie porty i protokoły kryją się pod daną nazwą — przydatne przed dodaniem usługi, której nie jesteśmy pewni, albo przy audycie „co naprawdę otwiera ta reguła".

Przykład:

```
http
  ports: 80/tcp
  protocols:
  source-ports:
  modules:
  destination:
  includes:
  helpers:
```

firewall-cmd = sterowanie zaporą firewalld

TYLKO ODCZYT

Wszystkie najważniejsze przełączniki

Przełącznik	Działa na	Efekt
(brak `--permanent`)	runtime	natychmiast, do testu, ginie po `--reload`/restarcie
`--permanent`	trwała konfiguracja	zapisuje, ale nie stosuje do ruchu na żywo
`--reload`	oba	wczytuje `--permanent` do runtime, kasując niezapisane zmiany runtime
`--runtime-to-permanent`	oba	zapisuje bieżący runtime jako nową konfigurację trwałą
`--timeout=Ns`	runtime	reguła sama znika po `N` sekundach — zabezpieczenie testowe
`--zone=nazwa`	dowolna komenda	kieruje operację do konkretnej strefy zamiast domyślnej

Uważaj

- `--add-port ... --permanent` **bez** `--reload` to reguła, która istnieje tylko na papierze — najczęstsza pomyłka na tej stronie.
- `--add-port ...` **bez** `--permanent` to reguła, która zniknie przy najbliższym `--reload` albo restarcie — dobra do testu, zła jako jedyny krok.
- Zmiana reguł dla SSH (usunięcie usługi, zmiana strefy domyślnej na `drop`) na zdalnej sesji może odciąć Cię od maszyny bez możliwości cofnięcia z tej samej sesji — testuj takie zmiany z `--timeout`.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera

Z życia wzięte

Z ŻYCIA WZIĘTE

Poniedziałek po Patch Tuesday. Nowa reguła na fw01 miała otworzyć port dla Graylog na docker01, ale ktoś dodał ją tylko w konfiguracji bieżącej, nie trwałej. Po restarcie usługi firewalld reguła znika, a Tomek uczy się na tym błędzie, żeby zawsze dodawać flagę permanent razem z reload.

Po co to istnieje

Pojęcie: zapora stanowa. Filtr ruchu sieciowego, który pamięta kontekst połączenia, a nie ocenia każdy pakiet z osobna. Gdy zezwolisz na wychodzące połączenie do portu 443, zapora stanowa automatycznie wpuści też pakiety powrotne tej samej rozmowy, bez osobnej reguły „wpuść odpowiedzi z portu 443" — rozpoznaje je jako część już zaakceptowanej sesji. `firewalld` (podobnie jak leżący pod nim `nftables` / `iptables`) działa właśnie w ten sposób; reguły, które piszesz, dotyczą głównie ruchu **inicjującego** nowe połączenia z zewnątrz.

Chcesz kontrolować, co wolno wejść na maszynę z sieci: który port, z jakiego źródła, jakim protokołem. `firewalld` istnieje jako warstwa nad `nftables` (a wcześniej `iptables`), która pozwala opisywać te reguły w sposób deklaratywny — strefami i nazwanymi usługami zamiast ręcznie pisanych łańcuchów reguł niskiego poziomu — i zmieniać je **na żywo**, przez wywołania D-Bus do działającego demona, bez przeładowywania całego zestawu reguł od zera przy każdej zmianie (co było typowe dla starych skryptów `iptables`).

To domyślna zapora na RHEL/Rocky i SLES. **Ubuntu domyślnie używa** `ufw` (Uncomplicated Firewall) — inny front-end nad tym samym mechanizmem jądra, z prostszą, ale mniej elastyczną składnią; `firewalld` jest na Ubuntu dostępny, ale trzeba go świadomie zainstalować i włączyć, wyłączając wcześniej `ufw`, żeby oba nie próbowały zarządzać tymi samymi regułami jądra naraz.

Analogia

`firewalld` to **recepcja z dwiema wersjami regulaminu**: tą, według której ochrona **faktycznie** działa dzisiaj, i tą spisaną w segregatorze, która wejdzie w życie **dopiero przy zmianie zmiany**.

Kiedy krzykniesz do ochroniarza „wpuszczaj też dostawców na bramę 443" bez zastrzeżenia „i zapisz to w regulaminie" — ochroniarz wpuszcza dostawców od razu, ale zapomni o tym, gdy zacznie się nowa zmiana (`--reload`). Kiedy zamiast tego dopiszesz zasadę do segregatora (`--permanent`) i wyjdziesz, nic się nie zmienia na bramie, dopóki ktoś nie ogłosi zmiany zmiany i nowy ochroniarz nie otworzy segregatora od nowa (`--reload`). Jedyny sposób, żeby coś zaczęło działać **i** przetrwało zmianę zmiany, to zrobić obie rzeczy: dopisać do segregatora i ogłosić zmianę zmiany.

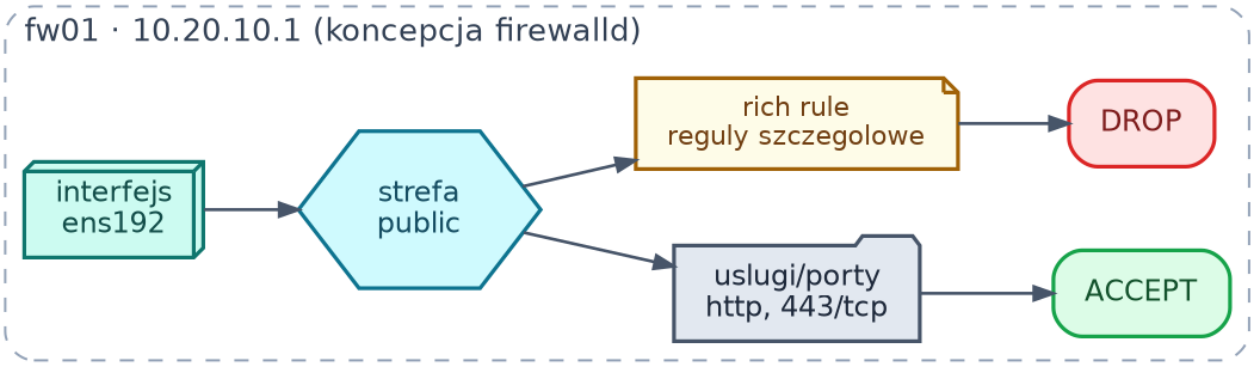
`--runtime-to-permanent` to skrót: „przepisz do segregatora dokładnie to, co ochroniarz robi teraz, bez pytania go o szczegóły" — przydatne, gdy już wypracowałeś żywą regułę metodą prób i błędów i nie chcesz jej teraz ręcznie odtwarzać.

Rich rule z `protocol value="vrrp"` to zasada napisana nie o konkretnej bramie (port), tylko o całym **rodzaju pojazdu** — „wpuszczaj każdy pojazd tego typu, niezależnie od tego, którą bramą wjeżdża" — bo VRRP nie rezerwuje sobie jednej konkretnej „bramy" (portu), tylko jeździ bezpośrednio po głównej drodze (warstwie IP).

Jak to działa

Rozwiń: jak to działa pod spodem

> **Pojęcie: konfiguracja bieżąca kontra trwała** (rozwinięcie z Fiszki). Runtime żyje w pamięci demona `firewalld` i to on filtruje ruch w danej chwili — pod spodem materializuje się jako zestaw reguł `nftables` (na RHEL/Rocky i SLES od kilku dużych wersji; wcześniej `iptables`). Permanent to pliki XML w `/etc/firewalld/zones/*.xml` i `/etc/firewalld/firewalld.conf` — czytane przy starcie usługi `firewalld.service` i przy każdym `--reload`.



Ruch trafia do strefy firewalld, a stamtąd do reguł usług i rich rules decydujących o dopuszczeniu. (kliknięcie otwiera pełny rozmiar)

Trzy operacje, które przenoszą dane między tymi dwoma światami:

- **start usługi** — permanent → runtime (budowanie stanu początkowego),
- `--reload` — permanent → runtime, **nadpisując** cokolwiek było w runtime i nie zostało zapisane jako permanent,
- `--runtime-to-permanent` — runtime → permanent, w drugą stronę, zapisując bieżący stan na dysk.

Nie ma żadnego automatycznego mechanizmu synchronizacji „w tle" — każda z tych operacji jest jawnym poleceniem. Stąd dwa lustrzane błędy, które opisuje reguła ze wstępu tej strony: reguła tylko-permanent, o której zapomniano zrobić `--reload` (nie działa, mimo że jest „dodana"), i reguła tylko-runtime, o której zapomniano, że zniknie przy najbliższym `--reload` (działała wczoraj, dziś nagle nie działa, bo ktoś zrobił `--reload` z innego powodu, np. po dodaniu zupełnie innej reguły).

> **Pojęcie: usługa firewalld jako nazwany zestaw portów** (rozwinięcie z Fiszki). Definicje usług leżą jako pliki XML w `/usr/lib/firewalld/services/*.xml` (systemowe, dostarczane z pakietami) i opcjonalnie `/etc/firewalld/services/*.xml` (własne, nadpisujące albo dodatkowe). Plik `http.xml` w skrócie mówi „port 80, protokół tcp" — `--add-service=http` to więc czysto składniowy skrót na `--add-port=80/tcp`, z tą przewagą, że w audycie reguł czytelne jest „http", a nie trzeba pamiętać numeru portu ani sprawdzać go w drugim źródle.

Rich rules rozszerzają ten model o warunki, których nie da się wyrazić samym portem czy usługą: konkretny adres źródłowy, konkretny protokół IP (jak VRRP, który nie ma portu), kierunek, logowanie, limit czasu. Składnia to pojedynczy ciąg tekstowy w cudzysłowie, budowany z opcjonalnych elementów w ustalonej kolejności: `rule [family="..."] [source ...]`

`[service name="..." | port port="..." protocol="..." | protocol value="..."] [log ...] [audit] accept|reject|drop`. Nawiasy w cudzysłowach wewnątrz jednej reguły to najczęstsze źródło błędów składniowych — patrz *Częste błędy*.

Krok po kroku

1. Sprawdź stan wyjściowy.

```
firewall-cmd --state
firewall-cmd --get-default-zone
sudo firewall-cmd --list-all
```

2. Przetestuj nową regułę na żywo, zanim ją utrwalisz.

```
sudo firewall-cmd --add-port=8080/tcp
curl -s -o /dev/null -w "%{http_code}\n" http://127.0.0.1:8080/
```

```
success
200
```

Reguła działa natychmiast, bez `--reload` — bo nie ma jeszcze `--permanent`, więc trafiła prosto do runtime.

3. Utrwal dokładnie to, co już przetestowałeś.

```
sudo firewall-cmd --runtime-to-permanent
sudo firewall-cmd --permanent --list-ports
```

```
success
8080/tcp
```

Zamiast ręcznie powtarzać `--add-port=8080/tcp --permanent`, `--runtime-to-permanent` przepisał do pliku trwałej konfiguracji cały bieżący stan runtime naraz.

4. Zweryfikuj, że reguła przetrwa `--reload`.

```
sudo firewall-cmd --reload
sudo firewall-cmd --list-ports
```

```
success
8080/tcp
```

Port jest nadal na liście po `--reload` — dowód, że reguła rzeczywiście trafiła do konfiguracji trwałej, a nie zniknęła.

5. Dodaj regułę o wyższym ryzyku z zabezpieczeniem `--timeout`.

```
sudo firewall-cmd --add-rich-rule='rule family="ipv4" source address="10.20.20.0/24" port
port="8080" protocol="tcp" reject' --timeout=60s
sleep 65
sudo firewall-cmd --list-rich-rules
```

```
success
```

Po upływie minuty lista rich rules jest znowu pusta — reguła sama się usunęła, dokładnie zgodnie z deklarowanym `--timeout`, bez potrzeby ręcznego sprzątania.

Częste błędy

`--permanent` bez `--reload` — reguła istnieje tylko na papierze.

```
sudo firewall-cmd --add-port=443/tcp --permanent
curl -v https://10.20.10.42/
```

```
success
curl: (7) Failed to connect to 10.20.10.42 port 443: Connection refused
```

Polecenie zwróciło `success`, port jest zapisany w konfiguracji trwałej — a mimo to połączenie jest odrzucane, bo runtime wciąż nie wie o nowym porcie. Naprawa: `sudo firewall-cmd --reload` zaraz po każdej zmianie z `--permanent`, którą trzeba zobaczyć na żywo od razu.

Reguła bez `--permanent` — działała wczoraj, dziś zniknęła.

```
sudo firewall-cmd --add-port=8080/tcp
# następnego dnia, po niepowiązanej zmianie z --reload:
curl http://127.0.0.1:8080/
```

```
curl: (7) Failed to connect to 127.0.0.1 port 8080: Connection refused
```

Reguła z pierwszego dnia nigdy nie trafiła do konfiguracji trwałej — zniknęła przy pierwszym `--reload`, niezależnie od tego, kto i po co go wywołał. Naprawa: każda reguła, która ma przetrwać, potrzebuje `--permanent` (i `--reload`) albo `--runtime-to-permanent`.

Zamknięcie sobie dostępu SSH przez zmianę strefy domyślnej.

```
sudo firewall-cmd --set-default-zone=drop --permanent
sudo firewall-cmd --reload
```

```
success
success
ssh: connect to host 10.20.10.42 port 22: Connection timed out
```

Strefa `drop` odrzuca cały ruch bez wyjątków — łącznie z SSH, którym akurat administrator jest podłączony. Po `--reload` sesja nie zrywa się natychmiast (istniejące połączenie może przetrwać chwilę jako już nawiązane), ale każda nowa próba połączenia jest blokowana, w tym reconnect po utracie sesji. Naprawa wymaga dostępu poza siecią (konsola/KVM) — dlatego ryzykowne zmiany strefy testuje się z `--timeout` albo z otwartą drugą sesją na wypadek błędu.

Reguła dodana do niewłaściwej strefy.

```
sudo firewall-cmd --zone=internal --add-port=443/tcp --permanent
sudo firewall-cmd --reload
curl https://10.20.10.42/
```

```
success
curl: (7) Failed to connect to 10.20.10.42 port 443: Connection refused
```

Port trafił do strefy `internal`, ale interfejs `ens192`, przez który przychodzi ruch, jest w strefie `public` — reguła istnieje, ale nigdy nie zostanie dopasowana do tego ruchu. Naprawa: `firewall-cmd --get-active-zones` pokazuje, która strefa faktycznie obsługuje dany interfejs, zanim dodasz do niej regułę.

Błąd składni w rich rule przez niedopasowane cudzysłowy.

```
sudo firewall-cmd --add-rich-rule='rule family="ipv4" source address=10.20.20.20 service
name="ssh" accept' --permanent
```

```
Error: INVALID_RULE: bad address: source
```

Wartość `address` musi być w cudzysłowie wewnątrz reguły (`address="10.20.20.20"`), tak samo jak `family` i `name` — brak cudzysłowu wokół adresu psuje parser rich rule. Naprawa: każda wartość

atrybutu w rich rule idzie w cudzysłowie, cała reguła — w apostrofach powłoki, żeby wewnętrzne cudzysłowy przetrwały interpretację przez `bash`.

□□ Ćwiczenia

> Wszystko poniżej testujesz na porcie `8080` (ten sam, którego używa serwer testowy z poprzedniej strony rozdziału) i z regułami zabezpieczonymi `--timeout`, więc nawet pomyłka nie zablokuje Ci trwale dostępu do maszyny. Wymaga `sudo`.

Ćwiczenie 1 — cykl test → utrwalenie → weryfikacja

```
cd /tmp
python3 -m http.server 8080 &
PID=$!

sudo firewall-cmd --add-port=8080/tcp
curl -s -o /dev/null -w "runtime: %{http_code}\n" http://127.0.0.1:8080/

sudo firewall-cmd --runtime-to-permanent
sudo firewall-cmd --reload
curl -s -o /dev/null -w "po reload: %{http_code}\n" http://127.0.0.1:8080/

sudo firewall-cmd --remove-port=8080/tcp --permanent
sudo firewall-cmd --reload
kill $PID
```

W którym momencie, gdybyś pominął `--runtime-to-permanent`, port przestałby odpowiadać?

Odpowiedź

Zaraz po `sudo firewall-cmd --reload` — bez zapisania reguły jako trwałej, `--reload` wczytałby konfigurację permanent sprzed dodania portu 8080 i nadpisał nią runtime, usuwając regułę, która chwilę wcześniej działała. Drugie `curl` zwróciłoby błąd połączenia zamiast kodu 200.

Ćwiczenie 2 — bezpieczny test reguły blokującej z `--timeout`

```
sudo firewall-cmd --add-rich-rule='rule family="ipv4" port port="8080" protocol="tcp" reject'
--timeout=30s
sudo firewall-cmd --list-rich-rules

cd /tmp && python3 -m http.server 8080 &
PID=$!
curl -s -o /dev/null -w "z aktywną regułą: %{http_code}\n" --max-time 3 http://127.0.0.1:8080/
|| echo "połączenie odrzucone, zgodnie z oczekiwaniem"

sleep 31
curl -s -o /dev/null -w "po wygaśnięciu: %{http_code}\n" http://127.0.0.1:8080/
kill $PID
```

Dlaczego ta metoda testowania jest bezpieczniejsza niż od razu dodawać taką regułę z `--permanent` ?

Odpowiedź

Reguła z `--timeout` usuwa się sama z runtime po podanym czasie, niezależnie od tego, czy administrator zdąży ją ręcznie cofnąć. Gdyby ta sama reguła (przez pomyłkę zbyt szeroka, np. pasująca też do portu SSH) trafiła od razu do `--permanent` i `--reload`, jedynym sposobem na jej cofnięcie po utracie dostępu byłby fizyczny dostęp do konsoli maszyny. `--timeout` daje siatkę bezpieczeństwa dokładnie na wypadek takiej pomyłki.

Pytania kontrolne

1. Dodałeś `firewall-cmd --add-service=http --permanent`, ale przeglądarka nadal nie może połączyć się z serwerem WWW. Jakiego jednego polecenia najpewniej zabrakło?

Odpowiedź

`firewall-cmd --reload`. `--permanent` zapisuje regułę do konfiguracji trwałej, ale nie stosuje jej do ruchu na żywo — bez `--reload` runtime nadal działa według starej konfiguracji.

2. Czym różni się `--reload` od `--runtime-to-permanent` pod względem kierunku, w którym płyną dane?

Odpowiedź

`--reload` kopiuje dane z konfiguracji trwałej (permanent) do bieżącej (runtime), nadpisując runtime i kasując wszystko, co było w runtime, a nie zostało zapisane jako permanent. `--runtime-to-permanent` robi odwrotnie — kopiuje bieżący stan runtime do konfiguracji trwałej, zapisując na dysk to, co aktualnie działa.

3. Dlaczego reguła dla protokołu VRRP musi być rich rule z `protocol value="vrrp"`, a nie zwykłym `--add-port`?

Odpowiedź

VRRP nie działa na porcie TCP ani UDP — to osobny protokół IP (numer 112), działający bezpośrednio na warstwie IP. `--add-port` przyjmuje wyłącznie parę numer/protokół transportowy (np. `443/tcp`), więc nie potrafi wyrazić „wpuść cały protokół IP numer 112" — do tego służy rich rule z elementem `protocol value="..."`, który operuje na poziomie protokołu IP, a nie portu.

4. Masz zamiar przetestować regułę, która może przypadkiem zablokować Twoje własne połączenie SSH do zdalnego serwera. Jakie jedno ustawienie w poleceniu `firewall-cmd` minimalizuje ryzyko trwałego zablokowania się?

Odpowiedź

`--timeout=Ns` — reguła z tym parametrem działa tylko w runtime i sama się usuwa po zadanym czasie, niezależnie od tego, czy administrator zdąży ją ręcznie cofnąć. To pozwala przetestować ryzykowną regułę z gwarancją, że nawet przy najgorszym scenariuszu dostęp wróci sam po upływie ustawionego czasu.

Powiązane: ip: adresy, trasy, interfejsy · NetworkManager i nmcli: konfiguracja trwała · Kto nasłuchuje i kto z kim gada: ss, lsof, netstat · SSH — klucze prywatne i publiczne · chrony — synchronizacja czasu (port NTP w zaporze)

Źródła: stare BookStack „Weryfikacja zapory sieciowej (firewalld) oraz dodanie wyjątków" · OneNote „Ogolne" (Sprawdzenie zapory oraz dodanie wyjątków, w tym reguła dla protokołu VRRP) · `man 1 firewall-cmd` · `man 5 firewalld.zones` · `man 5 firewalld.richlanguage` · `man 5 firewalld.service`

Revision #16
Created 2026-09-17 21:43:18 UTC by Tomasz Konieczny

Updated 2026-10-03 19:09:01 UTC by Tomasz Konieczny