SELinux: podstawy, tryby, konteksty

Poziom: średniozaawansowany

Dotyczy: RHEL/Rocky 9–10 (polityka targeted), Ubuntu (AppArmor jako odpowiednik), SLES (AppArmor domyślnie) zaznaczone

Czas: ~24 min

Usługa działa, port jest otwarty, a mimo to serwer odpowiada „Permission denied" tam, gdzie zwykłe uprawnienia plików mówią, że wszystko powinno działać. To prawie zawsze SELinux — druga, niezależna warstwa kontroli dostępu, która pyta o coś więcej niż tylko „czy ten użytkownik ma prawo do tego pliku".

Skróty użyte na tej stronie (8)

Skrót	Co znaczy
AVC	Access Vector Cache: wpis w dzienniku SELinuksa opisujący, jaką operację zablokowano i dlaczego
DAC	Direct Attach Copper: krótki miedziany kabel ze wtyczkami SFP na obu końcach, do łączenia urządzeń w tej samej szafie albo w sąsiednich
MAC	Media Access Control: unikalny adres fizyczny karty sieciowej, przypisany fabrycznie
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SELinux	Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
UID	User ID: numer użytkownika w Linuksie
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Fiszka

TYLKO ODCZYT

`getenforce` — bieżący tryb w jednym słowie

```
getenforce
```

Najkrótsza możliwa odpowiedź na pytanie „czy SELinux w ogóle coś blokuje w tej chwili".

Przykład:

```
Enforcing
```

SELinux = Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL · **getenforce** = pokaż tryb SELinux

TYLKO ODCZYT

`sestatus` — pełny obraz stanu

```
sestatus
```

Rozszerzona wersja `getenforce`: tryb bieżący, tryb skonfigurowany na starcie, nazwa i wersja załadowanej polityki.

Przykład:

```
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      33
```

TYLKO ODCZYT

`setenforce` — przełączenie trybu bez restartu

```
sudo setenforce 0
```

Przełącza natychmiastowo na tryb `Permissive` — SELinux przestaje blokować, ale nadal loguje, co by zablokował. `sudo setenforce 1` wraca do `Enforcing`. Zmiana działa tylko do najbliższego restartu i nie modyfikuje pliku konfiguracyjnego — do trwałej zmiany służy `/etc/selinux/config` niżej.

Przykład:

```
(brak wyjścia przy powodzeniu)
```

SELinux = Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL

TYLKO ODCZYT

`/etc/selinux/config` — tryb trwały

```
sudo vim /etc/selinux/config
```

```
SELINUX=enforcing
SELINUXTYPE=targeted
```

Wartość `SELINUX` przyjmuje `enforcing`, `permissive` albo `disabled` i obowiązuje od następnego restartu systemu — `setenforce` nie modyfikuje tego pliku, więc restart bez wcześniejszej zmiany tutaj wraca do poprzedniego trybu. Przejście na `disabled` wymaga restartu, a przy pierwszym powrocie do `enforcing`/`permissive` jądro musi **przeetykietować** cały system plików od zera (bo w trybie `disabled` żadne nowe pliki nie dostają poprawnych etykiet) — na dużym dysku ten pierwszy restart bywa zauważalnie dłuższy.

vim = Vi IMproved: ulepszony edytor vi

TYLKO ODCZYT

`ls -Z` — kontekst pliku

```
ls -Z /var/www/html/index.html
```

Dokłada do standardowego wyniku `ls` kontekst bezpieczeństwa SELinuksa przypisany do pliku.

Przykład:

```
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

□□ **ls** = list: wypisz zawartość katalogu

□□ TYLKO ODCZYT

`ps -Z` — kontekst procesu

```
ps -Z -C httpd
```

To samo dla działających procesów — pokazuje, w jakiej domenie SELinuksa faktycznie działa dany proces, nie tylko jakim użytkownikiem systemowym jest uruchomiony.

Przykład:

```
LABEL                             PID TTY          TIME CMD
system_u:system_r:httpd_t:s0     1234 ?        00:00:00 httpd
```

□□ **ps** = process status: pokaż procesy

□□ TYLKO ODCZYT

`id -Z` — kontekst bieżącej sesji

```
id -Z
```

Pokazuje kontekst SELinuksa przypisany bieżącemu użytkownikowi zalogowanemu — istotne przy polityce MLS/MCS, gdzie ten sam login może mieć różne poziomy w zależności od sesji.

Przykład:

```
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

□□ **id** = pokaż użytkownika, jego numer UID oraz grupy, do których należy

□□ TYLKO ODCZYT

`semanage fcontext -a` — trwałe przypisanie kontekstu do ścieżki

```
sudo semanage fcontext -a -t httpd_sys_content_t '/srv/www(/.*)?'
```

Dodaje regułę mówiącą „każdy plik pod `/srv/www` powinien mieć kontekst `httpd_sys_content_t`" do **bazy polityki**, nie do samych plików. Wzorzec `(/.*)?` obejmuje

zarówno sam katalog, jak i całą jego zawartość rekurencyjnie. Sama ta komenda niczego jeszcze nie zmienia na dysku — do zastosowania reguły służy `restorecon` niżej.

semanage = SELinux manage: zarządzanie regułami SELinux

TYLKO ODCZYT

`restorecon -Rv` — zastosowanie reguł kontekstu

```
sudo restorecon -Rv /srv/www
```

Odczytuje bazę reguł (w tym te dodane przez `semanage fcontext`) i faktycznie nadpisuje etykiety plików na dysku, żeby były zgodne z regułami. `-R` działa rekurencyjnie, `-v` wypisuje każdą zmienioną ścieżkę.

Przykład:

```
Relabeled /srv/www from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /srv/www/index.html from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

restorecon = restore context: przywróć etykiety SELinux

TYLKO ODCZYT

`chcon` — zmiana kontekstu natychmiast, ale tymczasowo

```
sudo chcon -t httpd_sys_content_t /srv/www/index.html
```

Zmienia kontekst pojedynczego pliku od razu, bez dotykania bazy reguł `semanage`. Wygodne do szybkiego testu — i dokładnie dlatego niebezpieczne jako rozwiązanie docelowe: pierwsze `restorecon` (uruchomione ręcznie, przez aktualizację pakietu albo przy przeetykietowaniu po zmianie trybu) przywróci poprzedni kontekst z bazy reguł, kasując zmianę wprowadzoną przez `chcon`.

TYLKO ODCZYT

`matchpathcon` — jaki kontekst *powinna* mieć dana ścieżka

```
matchpathcon /srv/www/index.html
```

Sprawdza, jaki kontekst przewiduje baza reguł dla wskazanej ścieżki, bez patrzenia na to, jaki kontekst plik ma faktycznie teraz — różnica między wynikiem tego polecenia a wynikiem `ls -Z` mówi wprost, czy plik wymaga `restorecon`.

Przykład:

```
/srv/www/index.html system_u:object_r:httpd_sys_content_t:s0
```

TYLKO ODCZYT

`getsebool -a` — lista wszystkich przełączników

```
getsebool -a | grep httpd
```

Wypisuje wartości logiczne (booleany) sterujące zachowaniem polityki — każdy to gotowy, nazwany „wyłącznik" dla konkretnego, przewidzianego przez twórców polityki scenariusza, bez potrzeby pisania własnych reguł.

Przykład:

```
httpd_can_network_connect --> off
httpd_can_sendmail --> off
httpd_enable_homedirs --> off
httpd_use_nfs --> off
```

getsebool = wypisuje wartość przełącznika SELinux, na przykład haproxy_connect_any, który pozwala usłudze łączyć się z dowolnym portem

TYLKO ODCZYT

`setsebool -P` — trwała zmiana przełącznika

```
sudo setsebool -P httpd_can_network_connect on
```

Włącza konkretny boolean trwale (`-P` — persistent, przetrwa restart). Bez `-P` zmiana obowiązuje tylko do restartu, tak jak `setenforce` bez modyfikacji pliku konfiguracyjnego. Ten konkretny przełącznik pozwala procesom `httpd` nawiązywać połączenia sieciowe wychodzące — potrzebne np. dla aplikacji webowej łączącej się z bazą danych na innym hoście.

setsebool = CLI: włącz albo wyłącz flagę logiczną (boolean) polityki SELinux, z opcją trwałego zapisu ustawienia flagą -P

TYLKO ODCZYT

`semanage port -l` — jakie porty są przypisane do jakich usług

```
semanage port -l | grep http_port_t
```

Polityka SELinuksa wie o usługach nie tylko przez nazwę procesu, ale też przez numer portu, na którym mają prawo nasłuchiwać — `http_port_t` obejmuje domyślnie porty 80, 443 i kilka innych.

Przykład:

```
http_port_t                    tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
```

semanage = SELinux manage: zarządzanie regułami SELinux

TYLKO ODCZYT

`semanage port -a` — dopisanie niestandardowego portu

```
sudo semanage port -a -t http_port_t -p tcp 8080
```

Dopisuje port 8080/tcp do listy portów, na których proces w domenie `httpd_t` ma prawo nasłuchiwać. Bez tego `httpd` skonfigurowany na porcie 8080 zostanie zablokowany przez SELinux, mimo że sam plik konfiguracyjny `httpd.conf` i firewall na to pozwalają — trzecia, niezależna warstwa kontroli, o której administratorzy migrujący usługę na inny port często zapominają.

SELinux = Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL · **semanage** = SELinux manage: zarządzanie regułami SELinux

TYLKO ODCZYT

`ausearch -m AVC` — znalezienie odmów SELinuksa

```
sudo ausearch -m AVC -ts recent
```

SELinux zapisuje każdą odmowę jako zdarzenie typu `AVC` (Access Vector Cache) w tym samym dzienniku audytu, z którego korzysta `auditd` — `-ts recent` ogranicza wynik do

ostatnich dziesięciu minut.

Przykład:

```
type=AVC msg=audit(1758184001.552:912): avc:  denied  { name_connect } for  pid=1234
comm="httpd" dest=3306 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:mysqld_port_t:s0 tclass=tcp_socket
```

AVC = Access Vector Cache: wpis w dzienniku SELinuksa opisujący, jaką operację zablokowano i dlaczego · **SELinux** = Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL · **ausearch** = przeszukaj log audytu jądra, tu pod kątem zdarzeń SELinux oznaczonych avc

TYLKO ODCZYT

`sealert` — czytelne wyjaśnienie odmowy

```
sudo sealert -a /var/log/audit/audit.log
```

Z pakietu `setroubleshoot-server` — zamienia surowy wpis `AVC` na opis w ludzkim języku i, co najważniejsze, na gotową sugestię naprawy (zwykle konkretny `semanage` albo `setsebool`).

Przykład:

```
SELinux is preventing httpd from name_connect access on the tcp_socket port 3306.

*****  Plugin catchall (100. confidence) suggests   **************************
If you want to allow httpd to connect to network
Then you must tell SELinux about this by enabling the 'httpd_can_network_connect'
boolean.
Do
setsebool -P httpd_can_network_connect 1
```

AVC = Access Vector Cache: wpis w dzienniku SELinuksa opisujący, jaką operację zablokowano i dlaczego

TYLKO ODCZYT

`audit2allow` — wygenerowanie własnego modułu polityki

```
sudo ausearch -m AVC -ts recent | audit2allow -M mojmodul
```

Analizuje zebrane odmowy `AVC` i generuje z nich gotowy plik polityki (`mojmodul.pp`) zezwalający dokładnie na to, co zostało odrzucone — ostateczność, gdy żaden istniejący boolean nie pokrywa danego przypadku, używana świadomie i po przeczytaniu wygenerowanej reguły, nie automatycznie.

Przykład:

```
****************** IMPORTANT ***********************
To make this policy package active, execute:


semodule -i mojmodul.pp
```

AVC = Access Vector Cache: wpis w dzienniku SELinuksa opisujący, jaką operację zablokowano i dlaczego · **ausearch** = przeszukaj log audytu jądra, tu pod kątem zdarzeń SELinux oznaczonych avc

TYLKO ODCZYT

`semodule -i` — załadowanie własnego modułu

```
sudo semodule -i mojmodul.pp
```

Instaluje moduł polityki wygenerowany przez `audit2allow` (albo pobrany z innego źródła) do aktywnej polityki SELinuksa.

TYLKO ODCZYT

`semanage login -l` — mapowanie loginów systemowych na użytkowników SELinuksa

```
semanage login -l
```

Pokazuje, jaki użytkownik SELinuksa (nie systemowy) zostaje przypisany każdemu loginowi systemowemu przy zalogowaniu — istotne przy politykach MLS/MCS z rozdzielonymi poziomami dostępu, rzadziej używane na typowym serwerze z polityką `targeted`.

Przykład:

```
Login Name           SELinux User         MLS/MCS Range        Service
__default__          unconfined_u         s0-s0:c0.c1023        *
root                 unconfined_u         s0-s0:c0.c1023        *
```

semanage = SELinux manage: zarządzanie regułami SELinux

ZMIENIA SYSTEM

Pakiety narzędziowe

```
sudo dnf install -y policycoreutils-python-utils setroubleshoot-server
```

`policycoreutils-python-utils` dostarcza `semanage` i `audit2allow`; bez niego na minimalnej instalacji RHEL/Rocky te polecenia po prostu nie istnieją. `setroubleshoot-server` dostarcza `sealert` i demona, który generuje czytelne opisy odmów w tle.

RHEL = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **dnf** = Dandified YUM: menedżer pakietów RHEL/Rocky

TYLKO ODCZYT

Wszystkie polecenia

Polecenie	Do czego służy
`getenforce` / `sestatus`	odczyt bieżącego trybu / pełny stan

Polecenie	Do czego służy
`setenforce 0\	1`
`/etc/selinux/config`	zmiana trybu trwała, wymaga restartu przy `disabled`
`ls -Z`, `ps -Z`, `id -Z`	odczyt kontekstu pliku / procesu / sesji
`semanage fcontext -a`	reguła kontekstu dla ścieżki, trwała
`restorecon -Rv`	zastosowanie reguł kontekstu na dysku
`chcon -t`	zmiana kontekstu natychmiast, nietrwała
`matchpathcon`	jaki kontekst *powinien* być wg reguł
`getsebool -a` / `setsebool -P`	odczyt / trwała zmiana przełącznika
`semanage port -l` / `-a`	odczyt / dopisanie portu do typu usługi
`ausearch -m AVC`	znalezienie odmów w dzienniku audytu
`sealert -a`	czytelne wyjaśnienie odmowy z sugestią naprawy
`audit2allow -M`	wygenerowanie modułu polityki z odmów

zmiana trybu bez restartu, nietrwała

Polecenie	Do czego służy
`semodule -i`	załadowanie własnego modułu polityki

Uważaj

- `chcon` znika po najbliższym `restorecon` — do trwałej zmiany zawsze `semanage fcontext`.
- Zmiana `SELINUX=disabled` w `/etc/selinux/config` wymaga restartu i sprawia, że przy najbliższym włączeniu z powrotem cały system plików musi zostać przeetykietowany od zera — proces potrafiący zauważalnie wydłużyć jeden konkretny restart.
- `audit2allow` generuje regułę zezwalającą dokładnie na to, co było zablokowane — łącznie z odmowami będącymi realnym atakiem, jeśli ktoś nieopatrznie „przepuści" przez nie wszystkie zebrane zdarzenia `AVC` bez przeczytania ich najpierw.

AVC = Access Vector Cache: wpis w dzienniku SELinuksa opisujący, jaką operację zablokowano i dlaczego

Z życia wzięte

Z ŻYCIA WZIĘTE

Czwartek, 13:00: Ola przenosi aplikację zespołu na nowy port 8080 na app01, firewall ma otwarty port, proces nasłuchuje, a mimo to żaden request z zewnątrz nie dochodzi. Pierwszy odruch to `setenforce 0`, co natychmiast rozwiązuje problem i budzi naszą czujność, bo to żadna naprawa. `sudo ausearch -m AVC -ts recent` pokazuje odmowę `name_connect`, a `semanage port -a -t http_port_t -p tcp 8080` naprawia to bez wyłączania SELinuksa.

Po co to istnieje

Pojęcie: kontrola dostępu obowiązkowa (MAC) kontra uznaniowa (DAC). Klasyczne uprawnienia plików Linuksa (`rwx`, właściciel, grupa) to kontrola dostępu **uznaniowa** (Discretionary Access Control, DAC) — nazwa bierze się stąd, że to właściciel pliku decyduje, komu nada dostęp, według własnego uznania. Jeśli plik ma uprawnienia `777`, każdy proces działający jako dowolny użytkownik może go czytać i zapisywać — system nie ma nic więcej

> do powiedzenia. SELinux dokłada drugą, **niezależną** warstwę: kontrolę dostępu **obowiązkową** (Mandatory Access Control, MAC), w której o dostępie decyduje centralnie zdefiniowana polityka, a nie właściciel pliku — nawet plik z uprawnieniami `777` i nawet proces działający jako root mogą zostać zablokowani, jeśli polityka SELinuksa tego nie przewiduje. Te dwie warstwy działają **obie naraz i niezależnie**: żądanie dostępu musi przejść zarówno przez klasyczne uprawnienia DAC, jak i przez politykę MAC, żeby się powiodło. To jest sedno, które trzeba zrozumieć przed czymkolwiek innym na tej stronie — każdy kolejny akapit o kontekstach i typach jest tylko rozwinięciem tego jednego rozróżnienia.

Powód istnienia tej drugiej warstwy: klasyczne DAC ufa procesowi w pełni, gdy tylko biegnie z odpowiednim UID. Jeśli serwer WWW zostanie przejęty przez atakującego (np. przez podatność w aplikacji), proces `httpd` nadal działa jako użytkownik `apache` i DAC pozwoli mu na wszystko, do czego ten użytkownik ma prawo — co bywa zaskakująco dużo. MAC ogranicza `httpd` do wąskiej, z góry zdefiniowanej domeny działania, niezależnie od tego, przez co proces został przejęty: nawet w pełni przejęty `httpd` nie odczyta `/etc/shadow`, bo polityka SELinuksa tego nie przewiduje dla domeny `httpd_t`, zupełnie niezależnie od uprawnień plikowych.

Analogia

DAC działa jak **klucz do własnego mieszkania** — właściciel mieszkania decyduje, komu zrobić dorobiony klucz, i to jego jedyne słowo w tej sprawie. MAC, czyli SELinux, działa jak **ochrona budynku z dodatkową, niezależną listą przepustek**, która obowiązuje bez względu na to, czyj to klucz.

Masz klucz do swojego mieszkania (uprawnienia plikowe pozwalają) — ale ochrona przy wejściu do budynku (SELinux) i tak sprawdza, czy Twoja przepustka pozwala Ci wejść tymi konkretnie drzwiami, o tej porze, z tym zadaniem. Recepcjonista nie pyta, czy masz klucz — zakłada, że tak, skoro w ogóle próbujesz wejść. Pyta o coś innego: czy Twoja rola w tym budynku (sprzątaczka, informatyk, gość) w ogóle przewiduje wejście do tego pomieszczenia. Sprzątaczka z kluczem pasującym fizycznie do drzwi serwerowni i tak nie zostanie wpuszczona, jeśli jej przepustka nie obejmuje tego pomieszczenia — dokładnie tak, jak proces `httpd` z formalnie poprawnymi uprawnieniami plikowymi do jakiegoś katalogu i tak zostanie zablokowany, jeśli polityka SELinuksa nie przewiduje dla jego domeny dostępu do plików o danym typie.

Kluczowy wniosek z analogii: wymiana zamka w drzwiach (`chmod`) nic nie zmienia w systemie przepustek ochrony. Odwrotnie też — rozdanie nowych przepustek (zmiana polityki SELinuksa) nic nie zmienia w tym, kto ma fizyczne klucze. Oba systemy muszą się zgodzić, każdy niezależnie.

Jak to działa

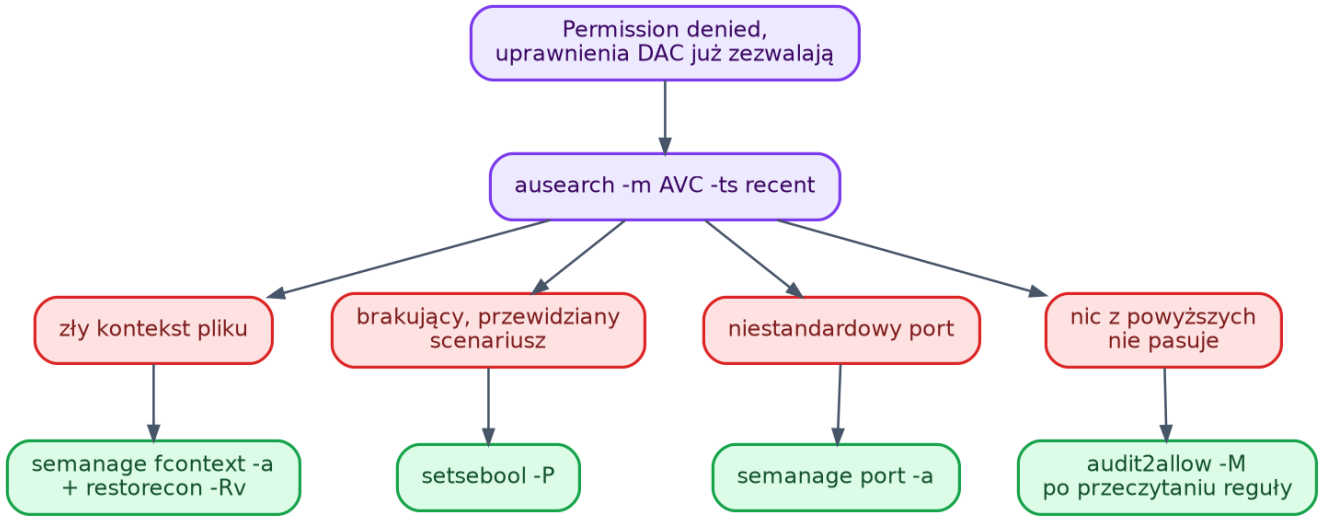
Rozwiń: jak to działa pod spodem

Pojęcie: kontekst bezpieczeństwa, typ i domena. Każdy plik, proces, port i inny obiekt w systemie z włączonym SELinuksem ma przypisany **kontekst bezpieczeństwa** — etykietę w formacie `użytkownik:rola:typ:poziom`, np. `system_u:system_r:httpd_t:s0`. W praktyce codziennej administracji trzecia część, **typ** (dla procesów nazywana **domeną**), robi niemal całą robotę: polityka `targeted` (domyślna na RHEL/Rocky) w zdecydowanej większości reguł porównuje właśnie typ pliku z domeną procesu, a nie użytkownika czy rolę. Proces `httpd` działa w domenie `httpd_t`; polityka zawiera regułę mówiącą, że domena `httpd_t` może czytać pliki typu `httpd_sys_content_t` — i to wystarcza, żeby serwer WWW obsłużył plik w `/var/www/html`, o ile ten plik faktycznie ma ten typ.

Pojęcie: polityka targeted. SELinux wspiera kilka modeli polityki; ten domyślnie włączony na RHEL/Rocky, Fedorze i pochodnych nazywa się `targeted` i celowo obejmuje ścisłą kontrolą tylko wybrane, „ucelowane" usługi systemowe działające w ograniczonych domenach (`httpd_t`, `sshd_t`, `named_t` i dziesiątki innych) — zwykłe procesy użytkownika działają w szerokiej, mało ograniczonej domenie `unconfined_t`. To wyjaśnia, dlaczego SELinux „przeszkadza" głównie przy usługach sieciowych, a rzadko przy codziennej pracy w powłoce: właśnie tam skupia się cała restrykcyjność polityki `targeted`.

Pojęcie: wartość logiczna (boolean). Wiele decyzji polityki jest sparametryzowanych przez nazwane przełączniki logiczne (booleany) — zamiast pisać własną regułę pozwalającą `httpd` na połączenia sieciowe wychodzące, wystarczy włączyć gotowy, przetestowany przez twórców polityki boolean `httpd_can_network_connect`. Booleany istnieją właśnie po to, żeby najczęstsze, przewidziane scenariusze (serwer WWW łączący się z bazą danych, serwer WWW obsługujący katalogi domowe użytkowników) dały się włączyć jednym poleceniem, bez ręcznego pisania i kompilowania modułów polityki.

Mechanizm sprawdzania w skrócie: gdy proces próbuje wykonać operację (odczyt pliku, połączenie sieciowe, uruchomienie innego programu), jądro pyta moduł SELinuksa, czy domena tego procesu ma w polityce regułę zezwalającą na tę operację względem typu docelowego obiektu. Jeśli reguły nie ma — operacja jest odrzucana (w trybie `Enforcing`) albo tylko logowana jako potencjalna odmowa (w trybie `Permissive`), niezależnie od wyniku klasycznego sprawdzenia uprawnień DAC, które zachodzi osobno. Każda taka odmowa (albo, w trybie permisywnym, „byłaby odmowa") trafia do dziennika audytu jako zdarzenie typu `AVC`.



Diagnostyka SELinuksa: od ausearch -m AVC do jednej z czterech celowanych napraw, zamiast wyłączania całej warstwy ochrony (kliknięcie otwiera pełny rozmiar)

Zasada praktyczna, którą warto wyprowadzić z tego mechanizmu wprost: **wyłączenie SELinuksa nie jest rozwiązaniem problemu, tylko rezygnacją z całej tej warstwy ochrony.** Właściwa procedura przy „Permission denied", które nie wynika z klasycznych uprawnień, wygląda tak: `ausearch -m AVC -ts recent`, żeby zobaczyć, co dokładnie zostało odrzucone i dlaczego → zrozumienie, czy chodzi o zły kontekst pliku (`semanage fcontext` + `restorecon`) czy o brakujący, przewidziany scenariusz (`setsebool`) → dopiero jako ostateczność, świadomie i po przeczytaniu wygenerowanej reguły, `audit2allow`. Usługa nasłuchująca na niestandardowym porcie potrzebuje `semanage port -a`, nie wyłączenia SELinuksa — port jest po prostu kolejnym typem obiektu, tak samo jak plik, i tak samo podlega tym samym regułom typ-do-typu.

Krok po kroku

1. Sprawdź bieżący tryb, zanim cokolwiek zaczniesz diagnozować.

```
getenforce
sestatus
```

2. Zdiagnozuj „Permission denied" krok po kroku, zamiast od razu wyłączać.

```
sudo ausearch -m AVC -ts recent
```

3. Jeśli masz `setroubleshoot-server`, przeczytaj gotową sugestię.

```
sudo sealert -a /var/log/audit/audit.log
```

4. Rozróżnij dwa różne rodzaje problemu: zły kontekst pliku czy brakujący boolean.

Zły kontekst pliku (typowe po przeniesieniu plików `mv` z lokalizacji o innym typie zamiast `cp`, który dziedziczy kontekst katalogu docelowego):

```
matchpathcon /srv/www/index.html
ls -Z /srv/www/index.html
```

Jeśli te dwa wyniki się różnią — to problem kontekstu, rozwiązywany przez `semanage fcontext -a` i `restorecon -Rv`, nigdy przez samo `chcon` na stałe.

Brakujący, przewidziany scenariusz (np. połączenie sieciowe wychodzące):

```
getsebool -a | grep network_connect
```

5. Dla usługi na niestandardowym porcie — dopisz port do właściwego typu.

```
sudo semanage port -a -t http_port_t -p tcp 8080
```

6. Dopiero gdy żadne z powyższych nie pasuje — wygeneruj własny moduł, świadomie.

```
sudo ausearch -m AVC -ts recent | audit2allow -M mojmodul
cat mojmodul.te
sudo semodule -i mojmodul.pp
```

Krok z `cat mojmodul.te` nie jest opcjonalny — to jest moment, w którym czytasz, na co dokładnie się zgadzasz, zanim to zainstalujesz.

Częste błędy

Wyłączenie SELinuksa jako pierwsza reakcja na „Permission denied".

```
sudo setenforce 0
```

Problem znika — bo cała warstwa ochrony przestaje działać, nie dlatego, że został rozwiązany. Serwer, który miał zostawać w wąskiej domenie `httpd_t`, teraz działa bez tego ograniczenia, i pozostaje tak do czasu, aż ktoś przypomni sobie, żeby to cofnąć — co często nie następuje nigdy. Naprawa: `ausearch -m AVC`, zrozumienie konkretnej odmowy, potem celowana poprawka (`semanage fcontext`, `setsebool` albo `semanage port`).

`chcon` traktowane jako trwałe rozwiązanie.

```
$ ls -Z /srv/www/index.html
system_u:object_r:var_t:s0 /srv/www/index.html
```

Administrator ręcznie ustawił kontekst przez `chcon -t httpd_sys_content_t`, serwer zaczął działać, sprawa uznana za zamkniętą. Kilka tygodni później, po rutynowym `restorecon -Rv /` wykonanym przy okazji innej naprawy, kontekst wraca do tego, co przewiduje baza reguł (tu — z powrotem `var_t`, bo nikt nie zaktualizował reguły przez `semanage fcontext`), i usługa znów odmawia dostępu, pozornie bez żadnej zmiany w konfiguracji. Naprawa: `semanage fcontext -a` zamiast samego `chcon` za każdym razem, gdy zmiana ma być trwała.

Usługa na niestandardowym porcie — próba naprawy przez firewall zamiast SELinux.

```
(usługa nie odpowiada na porcie 8080, mimo otwartego portu w firewalld)
```

Administrator otwiera port w `firewall-cmd`, sprawdza `ss -tlnp` — proces faktycznie nasłuchuje — a połączenia z zewnątrz i tak nie dochodzą albo sam proces nie potrafi w ogóle wystartować na tym porcie. Przyczyna: SELinux blokuje bindowanie do portu, który nie jest przypisany do typu portu zgodnego z domeną procesu (`http_port_t` dla `httpd_t`). Naprawa: `semanage port -a -t http_port_t -p tcp 8080`, dopiero po weryfikacji przez `ausearch -m AVC`, że to faktycznie SELinux, a nie inna warstwa, blokuje połączenie.

Zmiana trybu na `Disabled` bez świadomości kosztu przeetykietowania.

```
SELINUX=disabled
```

Administrator zmienia tryb w `/etc/selinux/config` na `disabled`, restartuje serwer, później zmienia zdanie i wraca do `enforcing`. Przy tym drugim restarcie jądro stwierdza, że etykiety plików mogły stać się nieaktualne (nowe pliki tworzone w trybie `disabled` nie dostawały żadnych kontekstów) i wymusza pełne przeetykietowanie całego systemu plików od zera — na serwerze z dużą ilością danych ten jeden restart może potrwać zauważalnie dłużej niż zwykle, czego nieświadomy administrator może pomylić z awarią.

Zapomnienie o `-P` przy `setsebool`.

```
sudo setsebool httpd_can_network_connect on
```

Boolean włącza się od razu, usługa zaczyna działać, problem wygląda na rozwiązany — do najbliższego restartu, po którym wartość wraca do domyślnej, bo bez `-P` zmiana nie została zapisana trwale. Naprawa: `setsebool -P`, zawsze, gdy zmiana ma przetrwać restart.

□□ Ćwiczenia

> Wszystko poniżej jest wyłącznie do odczytu — żadne polecenie nie zmienia trybu SELinuksa ani polityki hosta, na którym pracujesz. Jeśli chcesz przetestować `setenforce`, `semanage fcontext` albo `chcon` na żywo, zrób to na maszynie testowej.

Ćwiczenie 1 — odczyt bieżącego stanu bez żadnej zmiany

```
getenforce
sestatus
id -Z
```

Czy tryb zwrócony przez `getenforce` zgadza się z polem `Current mode` w `sestatus`? Które pole w `sestatus` mówi, jaki tryb zostanie przywrócony po najbliższym restarcie?

Odpowiedź

Oba pola powinny się zgadzać — `getenforce` to skrócona wersja tego samego odczytu, który `sestatus` pokazuje jako `Current mode`. Tryb po restarcie mówi pole `Mode from config file` — to ono odzwierciedla `/etc/selinux/config`, niezależnie od tego, czy ktoś w międzyczasie zmienił tryb tymczasowo przez `setenforce`.

Ćwiczenie 2 — kontekst pliku w katalogu tymczasowym

```
cd /tmp
touch plik-testowy.txt
ls -Z plik-testowy.txt
matchpathcon plik-testowy.txt
```

Porównaj oba wyniki. Czy plik utworzony w `/tmp` ma kontekst zgodny z tym, co przewiduje baza reguł dla tej lokalizacji? Dlaczego pliki w `/tmp` mają zwykle typ inny niż pliki w `/var/www`?

Odpowiedź

Powinny się zgadzać (typowo `tmp_t` dla katalogu `/tmp`) — nowe pliki dziedziczą domyślny kontekst dla swojej lokalizacji zgodnie z bazą reguł. Pliki w `/tmp` mają inny typ niż pliki w `/var/www`, bo polityka celowo rozróżnia „zwykłe pliki tymczasowe, do których dostęp mają niemal wszystkie domeny" od „treści serwowanej przez serwer WWW, do której dostęp ma mieć wyłącznie domena `httpd_t`" — to właśnie ta granica typów jest sensem MAC.

Ćwiczenie 3 — przegląd przełączników bez ich zmiany

```
getsebool -a | wc -l
getsebool -a | grep ' on$' | wc -l
getsebool -a | grep ssh
```

Ile jest w sumie booleanów dostępnych w tej polityce i ile z nich jest obecnie włączonych? Co to mówi o domyślnej postawie polityki `targeted` (raczej „domyślnie zamknięte" czy „domyślnie otwarte")?

Odpowiedź

Liczba całkowita zwykle sięga kilkuset (polityka `targeted` na pełnej instalacji RHEL/Rocky), a liczba włączonych jest zwykle wyraźnie mniejsza niż połowa. To odzwierciedla domyślną postawę „domyślnie zamknięte" — każdy dodatkowy, nietypowy scenariusz (połączenia sieciowe z nietypowych domen, dostęp do katalogów domowych) trzeba świadomie włączyć, zamiast że wszystko jest otwarte, dopóki ktoś czegoś nie zabroni.

Pytania kontrolne

1. Proces działa jako root i plik ma uprawnienia `777`, a mimo to dostęp jest odrzucony z komunikatem `Permission denied`. Jak to możliwe?

Odpowiedź

Bo uprawnienia klasyczne (DAC) i SELinux (MAC) to dwie niezależne warstwy kontroli, które muszą **obie** zezwolić na operację. Root i uprawnienia `777` satysfakcjonują warstwę DAC w pełni, ale nie mają żadnego wpływu na politykę MAC — jeśli domena procesu nie ma w polityce

SELinuksa reguły zezwalającej na dostęp do typu tego konkretnego pliku, operacja zostanie odrzucona niezależnie od uprawnień plikowych i tożsamości procesu.

2. Zmieniłeś kontekst pliku przez `chcon`, wszystko zaczęło działać, a po tygodniu przestało bez żadnej zmiany w konfiguracji. Co się prawdopodobnie stało?

Odpowiedź

Ktoś (albo jakiś proces systemowy, np. aktualizacja pakietu czy rutynowe `restorecon -R /`) uruchomił `restorecon`, który przywrócił kontekst zgodny z bazą reguł `semanage fcontext` — a ta baza nigdy nie została zaktualizowana, bo `chcon` zmienia kontekst tylko doraźnie, nie modyfikując reguł. Trwałe rozwiązanie wymagało `semanage fcontext -a` zamiast samego `chcon`.

3. Usługa nasłuchuje na porcie 8080 zamiast domyślnego 80, a `firewall-cmd --list-ports` pokazuje, że port jest otwarty. Mimo to połączenia nie dochodzą. Co sprawdzić?

Odpowiedź

Czy port 8080 jest przypisany do właściwego typu SELinuksa dla tej usługi — `semanage port -l | grep http_port_t` pokaże, czy 8080 jest już na liście. Jeśli nie, `semanage port -a -t http_port_t -p tcp 8080` dopisze go — firewall i SELinux to dwie osobne, niezależne bramki, obie muszą przepuścić ruch.

4. Dlaczego strona twierdzi, że wyłączenie SELinuksa „nie jest rozwiązaniem problemu"? Co jest właściwą alternatywą dla `setenforce 0` jako stałego rozwiązania?

Odpowiedź

Bo wyłączenie SELinuksa nie naprawia przyczyny odmowy — usuwa całą warstwę ochrony, przez co problem „znika" razem z zabezpieczeniem, które go zgłaszało. Właściwa ścieżka to diagnoza: `ausearch -m AVC` (i opcjonalnie `sealert`), żeby zrozumieć dokładnie, co i dlaczego zostało odrzucone, a potem celowana poprawka — zły kontekst naprawia `semanage fcontext` + `restorecon`, brakujący przewidziany scenariusz włącza `setsebool -P`, a niestandardowy port dopisuje `semanage port -a`.

Powiązane: Audyt i ślady logowań: auditd, last, lastb, journalctl · firewalld — strefy, usługi, reguły trwałe i bieżące · chmod i chown — uprawnienia klasyczne · sudo i /etc/sudoers: konfiguracja i audyt

Źródła: `man 8 selinux` · `man 8 semanage-fcontext` · `man 8 restorecon` · `man 8 audit2allow` · `man 8 sealert` · dokumentacja Red Hat: SELinux User's and Administrator's Guide

Revision #15
Created 2026-09-18 06:57:27 UTC by Tomasz Konieczny
Updated 2026-10-03 19:10:08 UTC by Tomasz Konieczny