# Codzienna praca

Wyszukiwanie, panele, alerty i reguły potoku.

- Graylog: codzienna praca z logami

# Graylog: codzienna praca z logami

**Poziom: średnio** **Dotyczy: Graylog 7.1.9 Open, OpenSearch 2.19.6, MailCow**

**Czas: ~60 min**

*Instalacja i wysyłka logów działają, więc zostaje to, co robimy w Graylogu na co dzień: pytamy o zdarzenia, dzielimy je na strumienie z osobną retencją, układamy panele i budujemy alerty z powiadomieniem e-mailem. Na końcu rozbijamy komunikat zapory na pola regułą potoku i porównujemy ją ze starszymi ekstraktorami.*

**Skróty użyte na tej stronie (9)**

| Skrót | Co znaczy |
|---|---|
| **AD** | Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie |
| **CA** | Certificate Authority: urząd certyfikacji, który podpisuje certyfikaty |
| **DNS** | Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP |
| **HTTP** | Hypertext Transfer Protocol: protokół stron WWW |
| **IT** | Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy |
| **LDAPS** | LDAP szyfrowany TLS, zwykle port 636 |
| **NAT** | Network Address Translation: podmiana adresów IP na granicy sieci |
| **SMTP** | Simple Mail Transfer Protocol: protokół wysyłania poczty |
| **STARTTLS** | włączenie szyfrowania w trakcie już nawiązanej, nieszyfrowanej sesji pocztowej |

# Fiszka

Wszystkie ścieżki w interfejsie podajemy po angielsku, tak jak w Graylogu 7.1. Producent przesuwa pozycje menu między wydaniami, więc w razie rozbieżności szukamy po nazwie. Zapytania wpisujemy w pole wyszukiwania na ekranie Search, a komendy wykonujemy na `docker01` (po przeprowadzce na `log01`) z uprawnieniami sudo.

TYLKO ODCZYT

## Wyszukanie nieudanych logowań w AD: EventID:4625

```
EventID:4625 AND TargetUserName:halina.kowalska
```

Zakres czasu ustawiamy na ostatnią godzinę, strumień na `ad-logowania`. Zdarzenie 4625 to nieudane logowanie w Windows, 4624 udane, 4740 zablokowanie konta. Nazwy pól zależą od agenta na `dc01` i `dc02` (NXLog i Winlogbeat nazywają je inaczej), więc przed pierwszym zapytaniem otwieramy jedną wiadomość i czytamy nazwy z listy pól. W tej stronie używamy nazw takich jak w dzienniku Windows.

Przykład:

```
2026-10-03 07:52:14.118  dc01  EventID=4625  TargetUserName=halina.kowalska
WorkstationName=RDS01  IpAddress=10.20.10.16
2026-10-03 07:52:44.201  dc01  EventID=4625  TargetUserName=halina.kowalska
WorkstationName=RDS01  IpAddress=10.20.10.16
```

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

TYLKO ODCZYT

## Wyszukanie odrzuconych połączeń na zaporze: fg_action:deny

```
fg_action:deny AND fg_dstport:22 AND NOT fg_srcip:10.20*
```

Odrzucone przez `fw01` próby połączenia na port 22, z pominięciem adresów z naszej sieci (gwiazdka na końcu wartości jest dozwolona, na początku nie). Pola z przedrostkiem `fg_` powstają dzięki regule z sekcji o potokach niżej.

Przykład:

```
2026-10-03 10:02:11.000  10.20.10.1  fg_srcip=203.0.113.77  fg_dstip=10.20.10.42
fg_dstport=22  fg_action=deny  fg_policyid=0
```

TYLKO ODCZYT

## Wyszukanie błędów aplikacji: level i source

```
source:app01 AND level:[0 TO 3]
```

Poziomy syslog od 0 (emerg) do 3 (err) z hosta `app01`. Nawias kwadratowy oznacza zakres z końcami włącznie, klamrowy wyklucza końce.

**□□ TYLKO ODCZYT**

## Wyszukanie frazy i pola, które istnieje: cudzysłów i _exists_

```
message:"Connection timed out" AND _exists_:fg_srcip
```

Fraza w cudzysłowie to dokładna sekwencja słów, `_exists_:nazwa_pola` zwraca tylko wiadomości, w których pole w ogóle występuje. Operatory `AND`, `OR`, `NOT` piszemy wielkimi literami, grupujemy nawiasami okrągłymi.

**□□ KLIKAMY W INTERFEJSIE**

## Ustawienie zakresu czasu: pasek zakresu na ekranie Search

```
Search → pasek zakresu czasu obok pola zapytania
→ Relative: ostatnie 5 minut, 1 godzina, 1 dzień
→ Absolute: od daty i godziny do daty i godziny
→ Keyword: opis słowny po angielsku, np. "yesterday", "last friday"
```

Zapytanie bez wąskiego zakresu czasu jest najczęstszą przyczyną wolnego wyszukiwania, bo Graylog przegląda wtedy wszystkie indeksy z zestawu.

**□□ ZRZUT EKRANU DO UZUPEŁNIENIA**

Zrzut do uzupełnienia: ekran Search z zapytaniem `EventID:4625`, rozwiniętym paskiem zakresu czasu (zakładka Absolute) i listą wiadomości poniżej.

**□□ KLIKAMY W INTERFEJSIE**

## Zapisanie wyszukiwania: Save na ekranie Search

```
Search → ikona zapisu (Save) → nazwa: "AD: nieudane logowania, ostatnia godzina"
Search → Saved Searches → wybór zapisanego widoku
```

Zapisane wyszukiwanie pamięta zapytanie, zakres czasu i wybrane widżety. Dajemy nazwy zaczynające się od źródła (`AD:`, `FW:`, `APP:`), żeby lista po pół roku dalej dało się czytać.

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

**ZMIENIA SYSTEM**

## Założenie zestawu indeksów z retencją: System → Indices

```
System → Indices → Create index set
→ Title: AD audyt
→ Index prefix: ad-audit
→ Index shards: 1, Index replicas: 0
→ Rotation/Retention: Data Tiering
→ Min days in storage: 180, Max days in storage: 181
```

Na `docker01` mamy jeden węzeł, więc replik nie ma sensu (nie mają gdzie wylądować). W Open dostępna jest warstwa gorąca bez archiwum i warstwy ciepłej, a odstęp między minimum a maksimum to zapas, w którym Graylog dobiera moment rotacji. Wartości dla pozostałych zestawów: `fw` 30 i 31 dni, `app` 30 i 31 dni.

**ZRZUT EKRANU DO UZUPEŁNIENIA**

Zrzut do uzupełnienia: formularz Create index set z wypełnionymi polami Title, Index prefix i sekcją Data Tiering (180 i 181 dni).

**ZMIENIA SYSTEM**

## Utworzenie strumienia: Streams → Create stream

```
Streams → Create stream
→ Title: ad-logowania
→ Index Set: AD audyt
→ Remove matches from 'Default Stream': zaznaczone
Streams → ad-logowania → Manage Rules → Add stream rule
→ Field: source, Type: match regular expression, Value: ^dc0
Streams → ad-logowania → Start Stream
```

Nowy strumień jest zatrzymany, dopóki go nie uruchomimy przyciskiem Start Stream. Analogicznie `fw-zapora`: reguła `source` pasuje do wyrażenia `^(fw01|10\.20\.10\.1)` (FortiGate bywa podpisany w nagłówku nazwą albo samym adresem, więc w pierwszej wiadomości sprawdzamy, co faktycznie trafia do pola `source`). I `app01-aplikacje`: `source` równe `app01`, zestaw indeksów `app`.

**◫ ZRZUT EKRANU DO UZUPEŁNIENIA**

Zrzut do uzupełnienia: ekran Streams z trzema strumieniami (`ad-logowania`, `fw-zapora`, `app01-aplikacje`), każdy w stanie Running, z liczbą wiadomości na sekundę.

**◫ ZMIENIA SYSTEM**

## Zbudowanie panelu: Dashboards → Create new dashboard

```
Dashboards → Create new dashboard → nazwa: "Zapora fw01"
→ Create → Aggregation
   Visualization: Bar chart, Group by: fg_srcip, Metrics: count()
   Query: fg_action:deny, Stream: fw-zapora
→ Save
```

Panel składa się z widżetów, a każdy widżet to zapytanie z agregacją: licznik, wykres słupkowy, tabela, lista wiadomości. Na `AD logowania` dokładamy wykres liczby zdarzeń 4625 w czasie, na `Zapora fw01` tabelę dziesięciu adresów z największą liczbą odrzuceń.

**◫ ZRZUT EKRANU DO UZUPEŁNIENIA**

Zrzut do uzupełnienia: panel „Zapora fw01” z wykresem słupkowym adresów `fg_srcip` i listą ostatnich odrzuceń.

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

**◫ PLIK KONFIGURACYJNY**

## Włączenie wysyłki e-mail w Graylogu: GRAYLOG_TRANSPORT_EMAIL_*

Plik `/docker/compose/graylog/compose.yaml`, w sekcji `environment` usługi `graylog` dokładamy zmienne. Nazwa zmiennej to ustawienie z `server.conf` pisane wielkimi literami z przedrostkiem `GRAYLOG_`, tak jak przy pozostałych ustawieniach w naszym Compose.

```
    environment:
      # ...dotychczasowe zmienne zostają...
      - GRAYLOG_TRANSPORT_EMAIL_ENABLED=true
      - GRAYLOG_TRANSPORT_EMAIL_HOSTNAME=poczta.zielona.example
```

```
      - GRAYLOG_TRANSPORT_EMAIL_PORT=587
      - GRAYLOG_TRANSPORT_EMAIL_USE_AUTH=true
      - GRAYLOG_TRANSPORT_EMAIL_USE_TLS=true
      - GRAYLOG_TRANSPORT_EMAIL_USE_SSL=false
      - GRAYLOG_TRANSPORT_EMAIL_AUTH_USERNAME=graylog@zielona.example
      - GRAYLOG_TRANSPORT_EMAIL_AUTH_PASSWORD=${SMTP_HASLO}
      - GRAYLOG_TRANSPORT_EMAIL_FROM_EMAIL=graylog@zielona.example
      - GRAYLOG_TRANSPORT_EMAIL_WEB_INTERFACE_URL=https://logi.zielona.example
```

Hasło skrzynki `graylog@zielona.example` zakładamy w panelu MailCow (opis w Instalacja MailCow na docker01 od zera) i wpisujemy do pliku `/docker/compose/graylog/.env` jako `SMTP_HASLO=...`, obok sekretów Grayloga. Port 587 z `USE_TLS=true` to STARTTLS, port 465 wymagałby `USE_SSL=true` i `USE_TLS=false`.

**STARTTLS** = włączenie szyfrowania w trakcie już nawiązanej, nieszyfrowanej sesji pocztowej

**ZMIENIA SYSTEM**

## Zastosowanie zmiany w Compose: docker compose up -d

```
cd /docker/compose/graylog
docker compose up -d graylog
```

Compose odtwarza tylko kontener `graylog`, MongoDB i OpenSearch zostają nietknięte. Odtworzony kontener traci zaufanie do firmowego CA dodane do magazynu `cacerts` komendą `keytool` w Graylog: logowanie kontem z Active Directory, więc po każdym odtworzeniu kontenera ten krok powtarzamy, inaczej logowanie LDAPS przestanie działać.

**CA** = Certificate Authority: urząd certyfikacji, który podpisuje certyfikaty · **LDAPS** = LDAP szyfrowany TLS, zwykle port 636 · **cd** = change directory: przejdź do katalogu · **docker** = narzędzie do kontenerów

**TYLKO ODCZYT**

## Sprawdzenie połączenia z MailCow: openssl s_client

```
openssl s_client -starttls smtp -connect poczta.zielona.example:587 -brief
```

Przykład:

```
CONNECTION ESTABLISHED
Protocol version: TLSv1.3
Peer certificate: CN = poczta.zielona.example
Verification: OK
```

Ten sam test wykonujemy z hosta, na którym stoi Graylog, bo to stamtąd pójdzie poczta. Jeżeli wewnętrzny DNS zwraca adres publiczny `203.0.113.10`, a `fw01` nie robi tak zwanego hairpin NAT, połączenie z `docker01` do samego siebie przez adres publiczny może nie przejść, wtedy dodajemy w wewnętrznym DNS wpis `poczta` na `10.20.10.42`.

**DNS** = Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP · **NAT** = Network Address Translation: podmiana adresów IP na granicy sieci · **openssl** = zestaw narzędzi do certyfikatów i szyfrowania

**ZMIENIA SYSTEM**

## Dodanie powiadomienia e-mail: Alerts → Notifications

```
Alerts → Notifications → Create notification
→ Title: E-mail do zespołu IT
→ Notification Type: Email Notification
→ Sender: graylog@zielona.example
→ Subject: Graylog: ${event_definition_title}
→ Email Recipient(s): it@zielona.example
→ Execute Test Notification
```

Przycisk Execute Test Notification wysyła próbną wiadomość od razu, więc ten krok zamyka pętlę z konfiguracją SMTP: jeżeli mail nie dojdzie, szukamy w logu kontenera `graylog`, nie w definicji zdarzenia.

**ZRZUT EKRANU DO UZUPEŁNIENIA**

Zrzut do uzupełnienia: formularz Create notification z typem Email Notification i wynikiem testu „Notification was executed successfully”.

**SMTP** = Simple Mail Transfer Protocol: protokół wysyłania poczty

**ZMIENIA SYSTEM**

## Dodanie definicji zdarzenia: Alerts → Event Definitions

```
Alerts → Event Definitions → Create event definition
→ Title: FW: skanowanie portów
→ Condition Type: Filter & Aggregation
→ Search Query: fg_action:deny AND fg_dstport:22
→ Streams: fw-zapora
→ Search within: 5 minutes, Execute search every: 5 minutes
→ Group by: fg_srcip
→ Create Events if: count() > 50
→ Notifications: E-mail do zespołu IT
→ Grace Period: 30 minutes
```

W wersji Open pracujemy na typie Filter & Aggregation: zapytanie plus licznik z warunkiem. Grace period to odstęp, w którym ten sam alert nie wyśle kolejnego powiadomienia, żeby jedno skanowanie nie zasypało skrzynki. Analogiczna definicja `AD: seria nieudanych logowań` ma zapytanie `EventID:4625`, grupowanie po `TargetUserName` i warunek `count() > 10` w oknie 5 minut.

**◫◫ ZRZUT EKRANU DO UZUPEŁNIENIA**

Zrzut do uzupełnienia: ostatni krok kreatora Event Definitions z podsumowaniem i podpiętym powiadomieniem e-mail.

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

**◫◫ KLIKAMY W INTERFEJSIE**

## Sprawdzenie kolejności procesorów wiadomości: System → Configurations

```
System → Configurations → Message Processors Configuration → Update
→ Message Filter Chain: przed Pipeline Processor
```

Message Filter Chain przypisuje wiadomości do strumieni według reguł strumieni (i uruchamia ekstraktory), Pipeline Processor uruchamia reguły potoków. Potok podpięty do strumienia zadziała tylko wtedy, gdy wiadomość jest już w tym strumieniu, czyli gdy Message Filter Chain stoi pierwszy. Kolejność domyślną sprawdzamy na własnej instalacji, nie zakładamy jej z pamięci.

**◫◫ KLIKAMY W INTERFEJSIE**

## Założenie potoku z regułą: System → Pipelines

```
System → Pipelines → Add new pipeline → nazwa: Zielona: zapora
→ Add new stage: Stage 0 (Messages that match: at least one rule)
→ Manage rules → Create rule (kod niżej)
→ Edit connections → strumień fw-zapora
```

Potok sam nic nie przetwarza, dopóki nie podepniemy go do strumienia. Etapy (stages) wykonują się po kolei, od najniższego numeru.

**TYLKO ODCZYT**

## Reguła potoku: rozbicie komunikatu FortiGate na pola

```
rule "FortiGate: pola z key=value"
when
  contains(to_string($message.message), "devname=")
then
  let pola = key_value(
    value: to_string($message.message),
    trim_value_chars: "\""
  );
  set_fields(fields: pola, prefix: "fg_");
end
```

Funkcja `key_value` tnie tekst po spacjach na pary `klucz=wartość`, `trim_value_chars` zdejmuje cudzysłowy z wartości, a `set_fields` zapisuje każdą parę jako pole wiadomości z przedrostkiem `fg_`. Przedrostek nie jest ozdobą: FortiGate wysyła własne pole `level="notice"` (tekst), a Graylog ma już pole `level` z liczbowym poziomem syslog, i zapis do tej samej nazwy kończy się konfliktem typów w indeksie.

Przykład komunikatu do wklejenia w symulator (System → Pipelines → Simulate processing):

```
date=2026-10-03 time=10:02:11 devname="fw01" devid="FG100FTK20000000" logid="0000000013"
type="traffic" subtype="forward" level="notice" vd="root" srcip=203.0.113.77
srcport=51544 dstip=10.20.10.42 dstport=22 proto=6 action="deny" policyid=0
service="SSH"
```

Po symulacji w wyniku mają stać pola `fg_srcip`, `fg_dstport`, `fg_action` i `fg_policyid`. Nie ufamy regule, której nie puściliśmy przez symulator na prawdziwej, świeżo zebranej

wiadomości.

**◫ ZRZUT EKRANU DO UZUPEŁNIENIA**

Zrzut do uzupełnienia: ekran Simulate processing z wklejonym komunikatem FortiGate po lewej i wynikiem z polami `fg_*` po prawej.

# Z życia wzięte

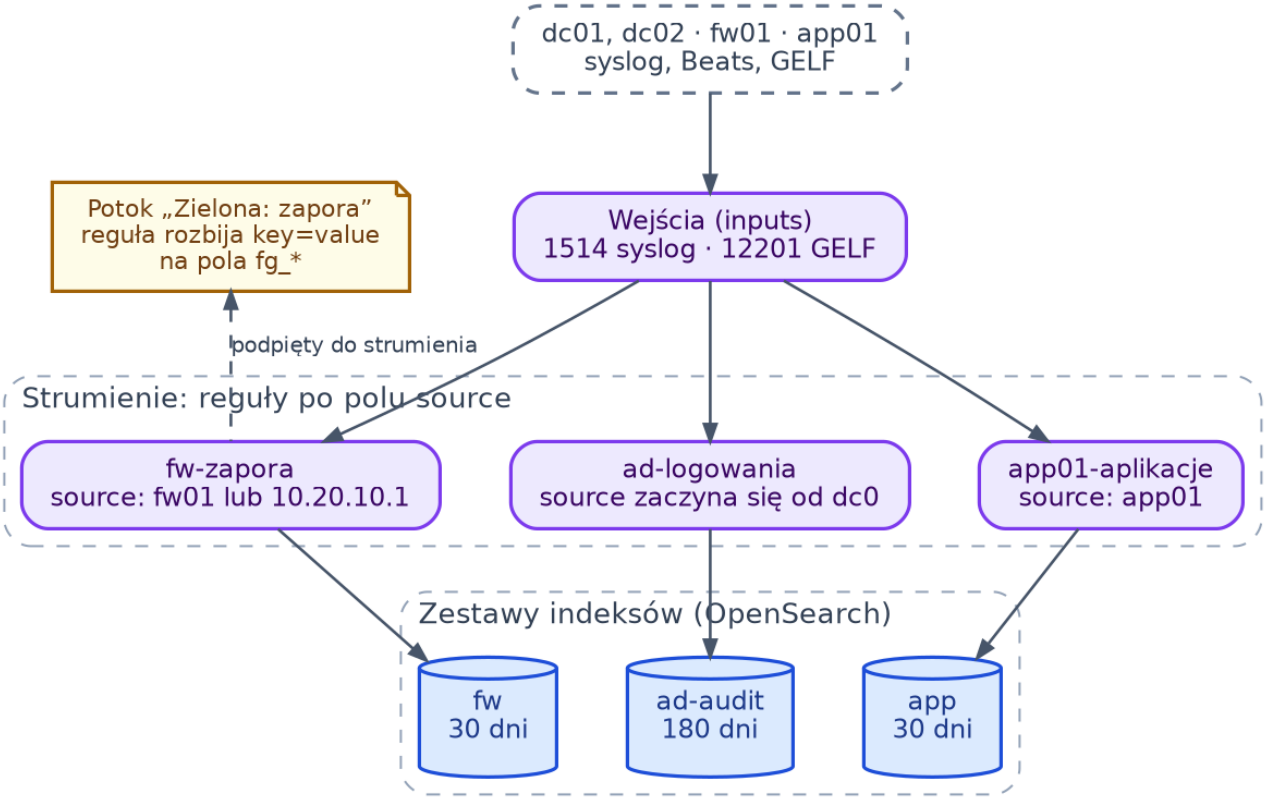
**◫ Z ŻYCIA WZIĘTE**

Środa, 7:55. Pani Halina dzwoni, że od rana jej konto domenowe blokuje się co kilka minut, mimo że hasła nie zmieniała. Tomek wpisuje `EventID:4625 AND TargetUserName:halina.kowalska`, ustawia zakres na dwie godziny i widzi regularny rytm: jedna nieudana próba co 30 sekund, zawsze ze stacji `RDS01`. Okazuje się, że rozłączona wczoraj sesja pani Haliny na `rds01` wciąż trzyma stare hasło. Po wylogowaniu sesji blokady znikają, a Tomek zapisuje to zapytanie jako `AD: nieudane logowania, ostatnia godzina` i dokłada do niego alert.

# Jak to działa

**◫ Rozwiń: jak to działa pod spodem**

Ścieżka wiadomości w Zielonej jest krótka: źródło wysyła log na wejście, reguły strumieni rozdzielają wiadomości do trzech strumieni, a każdy strumień ma własny zestaw indeksów z własną retencją. Potok, jeżeli jest podpięty, rozbija komunikat na pola, zanim wiadomość trafi do indeksu.




*Wiadomości z dc01, fw01 i app01 wchodzą na wejścia Grayloga, reguły strumieni rozdzielają je do trzech strumieni, potok rozbija komunikaty zapory na pola, a każdy strumień zapisuje do zestawu indeksów z własną retencją (kliknięcie otwiera pełny rozmiar)*

> **Pojęcie: strumień (stream).** Nazwany kontener na wiadomości wybrane regułą, na przykład wszystko z `fw01`. Strumień nie przechowuje danych sam, tylko kieruje wiadomości do zestawu indeksów, a na strumieniu opieramy zapisane wyszukiwania, panele i alerty.

> **Pojęcie: zestaw indeksów (index set).** Grupa indeksów OpenSearch z jedną wspólną retencją. Retencję ustawiamy na zestawie, nie na strumieniu, więc logi audytowe AD trzymane 180 dni i logi zapory trzymane 30 dni muszą leżeć w osobnych zestawach, a więc w osobnych strumieniach.

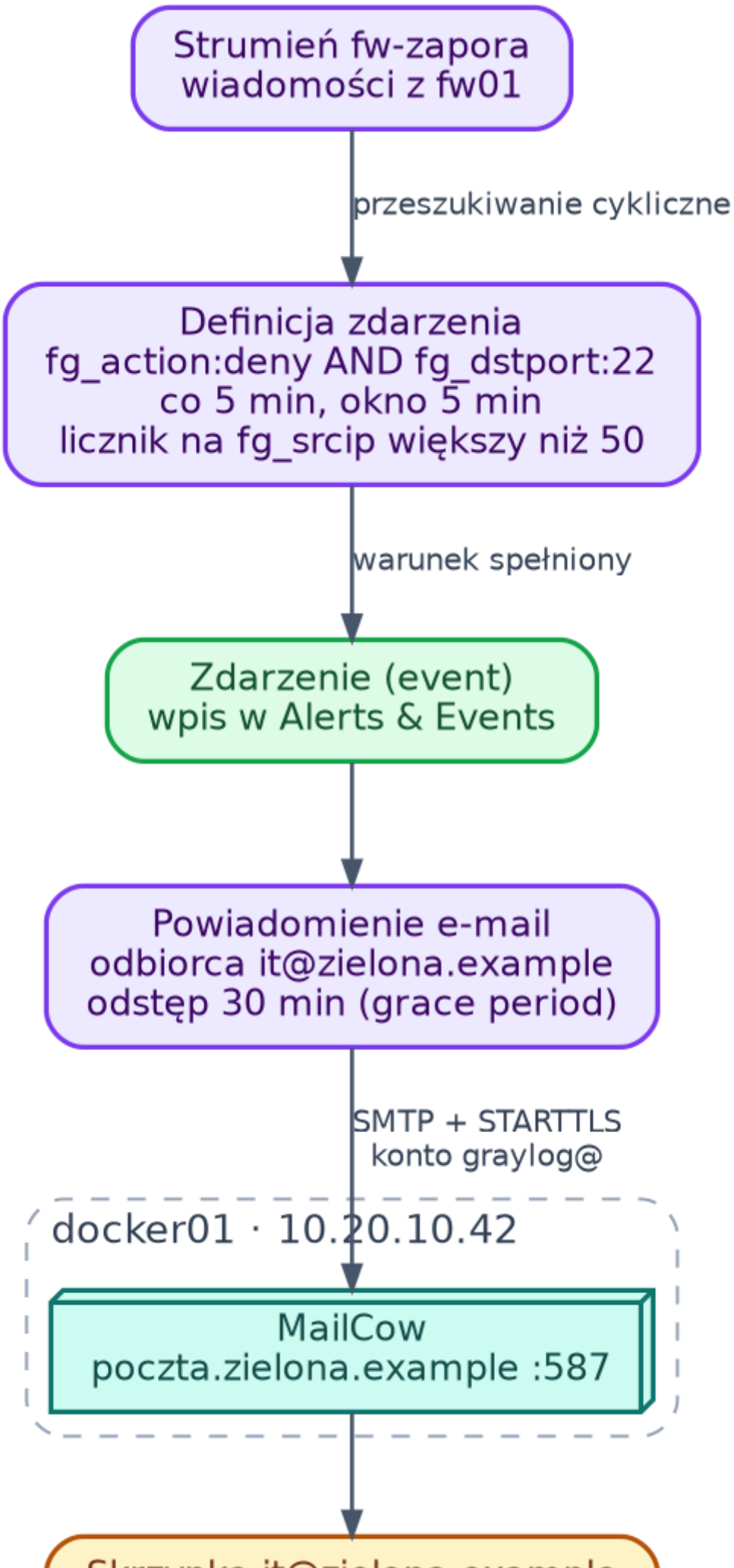
Jedna wiadomość trafia do wszystkich strumieni, których reguły spełnia, i jest zapisywana raz w zestawie każdego z nich. Dlatego strumienie, które nakładają się na siebie, kierujemy do tego samego zestawu indeksów, inaczej te same dane zajmą miejsce dwa razy. Wynik wyszukiwania w zapisanym widoku to nie osobna kopia danych, tylko zapytanie, więc `fg_action:deny` jest zapisanym wyszukiwaniem na strumieniu `fw-zapora`, a nie osobnym strumieniem.

# Panele, zdarzenia i powiadomienia

**Pojęcie: panel (dashboard).** Ekran z kilkoma widżetami, z których każdy jest zapytaniem z agregacją: licznik, wykres, tabela. Panel służy do patrzenia, nie do reagowania, więc wszystko, co ma nas zaalarmować, przenosimy do definicji zdarzenia.

**Pojęcie: definicja zdarzenia (event definition).** Zapytanie uruchamiane cyklicznie na wskazanych strumieniach, z warunkiem, na przykład liczba trafień większa niż 50. Gdy warunek jest spełniony, Graylog zapisuje zdarzenie (event) i wysyła powiadomienie.

**Pojęcie: powiadomienie (notification).** Sposób, w jaki zdarzenie dociera do człowieka: e-mail, HTTP albo komunikator. Powiadomienie e-mail korzysta z serwera pocztowego ustawionego zmiennymi `GRAYLOG_TRANSPORT_EMAIL_*`, w Zielonej z MailCow na `docker01`.


Strumień fw-zapora
wiadomości z fw01
przeszukiwanie cykliczne
Definicja zdarzenia
fg_action:deny AND fg_dstport:22
co 5 min, okno 5 min
licznik na fg_srcip większy niż 50
warunek spełniony
Zdarzenie (event)
wpis w Alerts & Events
Powiadomienie e-mail
odbiorca it@zielona.example
odstęp 30 min (grace period)
SMTP + STARTTLS
konto graylog@
docker01 · 10.20.10.42
MailCow
poczta.zielona.example :587

*Strumień fw-zapora jest cyklicznie przeszukiwany przez definicję zdarzenia, spełniony warunek tworzy zdarzenie, a powiadomienie e-mail trafia przez MailCow do skrzynki zespołu IT (kliknięcie otwiera pełny rozmiar)*

Rozróżnienie jest istotne w rozmowie: zdarzenie to fakt zapisany w Graylogu, alert to powiadomienie o nim. Zdarzenie bez powiadomienia też ma sens, bo widać je w Alerts → Alerts & Events i nadaje się do przeglądu raz dziennie.

# Reguły potoku kontra ekstraktory

> **Pojęcie: ekstraktor (extractor).** Reguła przypięta do jednego wejścia, która wycina pole z komunikatu wyrażeniem regularnym, wzorcem Grok albo podziałem po separatorze. Na stronie Graylog: wejścia, strumienie, wyszukiwanie nazywamy je rozszerzeniami. Dokumentacja producenta opisuje ekstraktory jako funkcję starszego typu (legacy) i zaleca potoki do nowych wdrożeń.

> **Pojęcie: reguła potoku (pipeline rule).** Fragment kodu w prostym języku Grayloga w postaci `when` (warunek) i `then` (działania), umieszczony na etapie potoku, który jest podpięty do strumienia. W przeciwieństwie do ekstraktora reguła ma warunki, funkcje i wiele działań naraz.

| Cecha | Ekstraktor | Reguła potoku |
|---|---|---|
| Do czego jest przypięty | do jednego wejścia | do strumienia |
| Warunek wykonania | brak, można tylko ograniczyć wyrażeniem | dowolny w części `when` |
| Co umie | wyciąć jedno pole z jednego pola | wiele pól, konwersje, routing, usuwanie wiadomości |
| Test przed wdrożeniem | podgląd na wiadomości z wejścia | symulator na wklejonej wiadomości |
| Zalecenie producenta | starszy typ (legacy) | zalecany do nowych wdrożeń |

Jak wygląda ta sama robota w obu światach, pokazuje nginx. Ekstraktor Grok na wejściu przyjmującym logi nginx używa wzorca `%{COMBINEDAPACHELOG}`. Reguła potoku robi to w jednym wierszu:

```
rule "nginx: pola z access.log"
when
  has_field("message") && contains(to_string($message.message), "HTTP/")
```

```
then
  set_fields(grok(pattern: "%{COMBINEDAPACHELOG}", value: to_string($message.message),
only_named_captures: true), prefix: "ng_");
end
```

Wzorzec `COMBINEDAPACHELOG` jest wśród domyślnych wzorców Grok, a listę widzimy w System → Grok Patterns. Format logu Nginksa musi zgadzać się z formatem `combined`, inaczej reguła po cichu niczego nie wyciągnie, o czym więcej w Graylog: częste problemy. W Zielonej nowych ekstraktorów nie zakładamy, a istniejące przepisujemy na potoki przy okazji, nie hurtem.

# Częste błędy

| Objaw | Przyczyna | Co robimy |
|---|---|---|
| Strumień jest pusty, choć wiadomości widać w `All messages` | strumień nie został uruchomiony albo reguła nie pasuje do pola `source` | Streams → strumień → Start Stream, w pierwszej wiadomości sprawdzamy, co jest w `source` |
| Zapytanie `source:*01` zwraca błąd | wiodące wildcardy są domyślnie wyłączone ( `allow_leading_wildcard_searches = false` ) | zapytanie z gwiazdką na końcu ( `source:dc0*` ) albo wyrażenie regularne `source:/.*01/` |
| Zapytanie po polu `fg_dstport:[1 TO 1024]` zwraca dziwne wyniki | `key_value` zapisuje wartości jako tekst, więc zakres liczy się alfabetycznie | porównujemy dokładne wartości albo dopisujemy konwersję liczbową w regule i sprawdzamy w symulatorze |
| Reguła potoku działa w symulatorze, ale na żywo pól nie ma | potok niepodpięty do strumienia albo Pipeline Processor stoi przed Message Filter Chain | System → Pipelines → Edit connections, System → Configurations → Message Processors |
| Alert przychodzi co 5 minut przez godzinę | brak grace period albo zbyt szeroki warunek | grace period 30 minut, próg w warunku ( `count() > 50` ), a nie samo trafienie |
| Test powiadomienia kończy się błędem, maila brak | `GRAYLOG_TRANSPORT_EMAIL_ENABLED` nie ustawione albo zły port i tryb TLS | `docker compose logs graylog` , port 587 z `USE_TLS=true` , test `openssl s_client` |
| Po odtworzeniu kontenera znika logowanie LDAPS | certyfikat CA był dodany do `cacerts` wewnątrz starego kontenera | powtarzamy krok z `keytool` ze strony o logowaniu AD |
| Ta sama wiadomość zajmuje dwa razy miejsce | trafia do dwóch strumieni z różnymi zestawami indeksów | strumienie nakładające się na siebie kierujemy do jednego zestawu |

# □□ Ćwiczenia

1. Budujemy zapytanie, które znajdzie nieudane logowania (4625) do kont kończących się na `.admin`, ale tylko ze stacji innych niż `DC01`. Operatory piszemy wielkimi literami.

**Odpowiedź**

Wiodące wildcardy są wyłączone, więc nie możemy napisać `*.admin`. Używamy wyrażenia regularnego i negacji: `EventID:4625 AND TargetUserName:/.*\.admin/ AND NOT WorkstationName:DC01`. Jeżeli wyrażenia regularne na danym polu nie działają (pole jest analizowane), zawężamy zakres czasu i filtrujemy po loginie wprost.

1. Strumień `fw-zapora` ma reguły poprawne, jest uruchomiony, a mimo to pola `fg_*` nie powstają. Wymieniamy dwie rzeczy do sprawdzenia, zanim zaczniemy poprawiać regułę.

**Odpowiedź**

Po pierwsze, czy potok jest podpięty do strumienia `fw-zapora` (Edit connections). Po drugie, czy w Message Processors Configuration Message Filter Chain stoi przed Pipeline Processor, bo inaczej potok sprawdza wiadomość, zanim trafi do strumienia. Dopiero potem wklejamy świeżą wiadomość do symulatora i patrzymy, czy warunek `when` jest prawdziwy.

1. Chcemy, żeby logi audytowe AD żyły 180 dni, a logi zapory 30. Wyjaśniamy, dlaczego nie wystarczy jeden strumień z dwiema regułami retencji.

**Odpowiedź**

Retencję ustawia się na zestawie indeksów, a strumień tylko wskazuje, do którego zestawu trafiają jego wiadomości. Dwa różne okresy przechowywania wymagają więc dwóch zestawów indeksów, a co za tym idzie dwóch strumieni, które do nich kierują.

**Powiązane:** Graylog: wejścia, strumienie, wyszukiwanie · Wysyłka logów z hostów · Graylog: logowanie kontem z Active Directory · Instalacja MailCow na docker01 od zera · Graylog: częste problemy

**Źródła:** go2docs.graylog.org: „Search syntax reference", „Extractors", „Stream Connections", „Index Model", „Data Tiering", „Alerts", „Event Definitions", „Graylog and Docker", „Additional Configuration" (sprawdzone 03.10.2026) · Graylog `server.conf` w gałęzi 7.1 (ustawienia `transport_email_*`) · community.graylog.org, wątki o kolejności procesorów wiadomości i o `key_value` w potokach · docs.fortinet.com, FortiOS Log Message Reference