Centralne logowanie

Po co agregujemy logi i jak działa Graylog.

- Po co centralne logowanie i jak to działa
- Graylog: wejścia, strumienie, wyszukiwanie

Po co centralne logowanie i jak to działa

Poziom: podstawy **Dotyczy: dowolne środowisko z więcej niż kilkoma serwerami**

Czas: ~15 min

Zanim zaczniemy konfigurować konkretne narzędzie, ustalamy, jaki problem centralne logowanie w ogóle rozwiązuje — bo to nie jest kolejny system do przeglądania plików, tylko odpowiedź na jedną konkretną awarię.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
GELF	Graylog Extended Log Format: format logów Grayloga
HTTP	Hypertext Transfer Protocol: protokół stron WWW
JSON	JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami
RFC	Request for Comments: oficjalny dokument opisujący standard internetowy, na przykład sposób działania danego protokołu; w ITIL także Request for Change, czyli wniosek o zmianę
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Z życia wzięte

Z ŻYCIA WZIĘTE

O 3:20 w nocy pada `app01`, akurat w środku wdrożenia, a Tomek loguje się rano i chce sprawdzić, co się działo tuż przed awarią. Lokalny `/var/log` na tym hoście jest pusty, bo dysk systemowy padł razem z partycją logów, więc jedyny ślad zdarzenia leżałby już tylko w centralnym systemie logowania, gdyby ten host w ogóle do niego wysyłał. Po tym incydencie Tomek dopisuje `app01` do listy hostów objętych wysyłką logów, żeby kolejna taka awaria nie zostawiła zespołu bez żadnego zapisu.

Po co to istnieje

> **Pojęcie: agregacja logów.** Zbieranie zdarzeń zapisywanych przez systemy operacyjne, usługi i aplikacje z wielu maszyn w jedno wspólne, przeszukiwalne miejsce, zamiast pozostawiania ich rozproszonych po lokalnych plikach na każdym hoście z osobna.

Log leżący wyłącznie lokalnie na maszynie, która właśnie padła, jest bezużyteczny dokładnie w chwili, w której jest najbardziej potrzebny — dysk systemowy uszkodzony razem z partycją `/var/log`, maszyna wirtualna usunięta razem z całym środowiskiem po nieudanej migracji, albo po prostu serwer, do którego straciliśmy dostęp SSH w środku incydentu, unieważniają jedyną kopię zapisu tego, co się na nim działo. Drugi, niezależny od losu monitorowanej maszyny problem to skala: przeszukanie logu z jednego zdarzenia rozłożonego na dwadzieścia serwerów aplikacyjnych za load balancerem oznacza dwadzieścia osobnych sesji SSH i dwadzieścia osobnych `grep`, zamiast jednego zapytania w jednym miejscu.

> **Pojęcie: centralne logowanie.** Architektura, w której logi opuszczają maszynę źródłową niemal natychmiast po ich wygenerowaniu i trafiają na dedykowany, oddzielny system — dzięki temu log przeżywa awarię maszyny, która go wyprodukowała, i staje się przeszukiwalny wspólnie z logami wszystkich pozostałych hostów.

Droga logu: źródło → transport → indeks → wyszukiwanie

Cztery etapy poniżej to szkielet każdego systemu centralnego logowania, niezależnie od konkretnego produktu — Graylog, opisany szerzej na stronie Graylog: wejścia, strumienie, wyszukiwanie, realizuje je wszystkie, ale te same cztery kroki znajdziemy też w ELK/OpenSearch Stack czy w rozwiązaniach chmurowych.



Log przechodzi przez cztery etapy: źródło, transport, indeks w silniku wyszukiwania i dopiero na końcu wyszukiwanie (kliknięcie otwiera pełny rozmiar)

Źródło. Proces generujący zdarzenie — demon `rsyslogd`/`journald` systemu operacyjnego, serwer aplikacyjny piszący własny format logu, urządzenie sieciowe (firewall, przełącznik) wysyłające komunikaty syslog natywnie.

> **Pojęcie: transport.** Protokół i format, jakim log opuszcza źródło i dociera do centralnego zbieracza. Trzy najczęściej spotykane:

Transport	Charakterystyka
syslog (UDP/TCP)	najstarszy, najszerzej wspierany natywnie przez systemy i urządzenia sieciowe; wariant UDP nie gwarantuje dostarczenia
GELF (Graylog Extended Log Format)	format JSON zaprojektowany pod Graylog, niesie ustrukturyzowane pola zamiast płaskiego tekstu, wspiera kompresję i dzielenie dużych komunikatów na części
Beats (Filebeat, Winlogbeat)	lekkie agenty czytające pliki albo dziennik zdarzeń Windows i wysyłające je dalej, z własnym potwierdzaniem dostarczenia

> **Pojęcie: indeks.** Ustrukturyzowana, przeszukiwalna reprezentacja logów wewnątrz silnika wyszukiwania (OpenSearch/Elasticsearch pod Graylogiem) — surowy tekst komunikatu trafia tu podzielony na pola, dzięki czemu wyszukiwanie po `srcip:10.20.10.42` jest operacją na indeksie, a nie linowym przeszukiwaniem tekstu.

Wyszukiwanie. Interfejs (webowy w Graylogu) pozwalający budować zapytania po polach, zapisywać je jako gotowe widoki i wykresy, oraz — co najistotniejsze operacyjnie — definiować alerty uruchamiane automatycznie, gdy zapytanie zacznie zwracać wyniki spełniające zadany warunek.

Model push — kto decyduje o wysyłce

> **Pojęcie: model push.** Centralne logowanie działa odwrotnie niż większość monitoringu opisanego na stronie Architektura Zabbixa: serwer, proxy, agent, frontend — to źródło samo, z własnej inicjatywy, wysyła zdarzenie w momencie jego wystąpienia, zamiast czekać, aż

centralny system zapyta o nowe dane. Ta różnica ma bezpośrednią konsekwencję dla firewalla: kierunek ruchu jest zawsze od hosta źródłowego do serwera logowania, nigdy odwrotnie, więc otwieramy ruch wychodzący z każdego monitorowanego hosta, a nie ruch przychodzący do niego.

Uważaj: model push oznacza też, że host, który padł w sposób uniemożliwiający mu jeszcze wysłanie ostatnich kilku sekund logów przed awarią (np. nagły restart bez czasu na dokończenie zapisu do kolejki), zabiera ten fragment ze sobą — centralne logowanie zabezpiecza przed utratą logów sprzed awarii, ale nie przed utratą tych dosłownie ostatnich, niezapisanych jeszcze zdarzeń.

Zdarzenie ustrukturyzowane kontra nieustrukturyzowane

Pojęcie: log nieustrukturyzowany. Pojedyncza linia tekstu bez z góry narzuconego podziału na pola — typowy klasyczny komunikat syslog, czytelny dla człowieka wprost, ale wymagający dodatkowego parsowania (przez rozszerzenia, opisane na stronie Graylog: wejścia, strumienie, wyszukiwanie), zanim stanie się przeszukiwalny po konkretnych wartościach, a nie tylko po dopasowaniu fragmentu tekstu.

Pojęcie: log ustrukturyzowany. Zdarzenie niosące już przy wysyłce podział na nazwane pola, zwykle w formacie JSON (jak GELF) — aplikacja pisząca log ustrukturyzowany decyduje raz, przy pisaniu kodu, jak nazwać pole `user_id` czy `request_duration_ms`, i od tego momentu każde wystąpienie tego pola jest przeszukiwalne wprost, bez etapu domyślania się formatu przez wyrażenie regularne po stronie centralnego systemu.

Kompromis między tymi dwoma podejściami jest praktyczny, nie ideologiczny: log nieustrukturyzowany jest tańszy do wyprodukowania (aplikacja pisze zwykły `print`/`printf`), ale droższy do utrzymania po stronie centralnego systemu (każda zmiana formatu źródłowego wymaga poprawki reguły parsowania). Log ustrukturyzowany wymaga świadomej decyzji przy pisaniu aplikacji, ale eliminuje tę kruchość.

Retencja i koszt

> **Pojęcie: retencja logów.** Okres, przez jaki logi pozostają dostępne do przeszukania, zanim zostaną usunięte albo przeniesione do tańszego, wolniej dostępnego magazynu. Retencja centralnego systemu logowania jest osobnym ustawieniem od retencji logów lokalnych na samych maszynach, opisanej dla `journald` na stronie Retencja logów — w typowej architekturze logi lokalne trzymamy krótko (kilka–kilkanaście dni, jako bufor na wypadek chwilowej niedostępności centralnego systemu), a to centralny system logowania odpowiada za dłuższą, uzgodnioną z wymaganiami audytowymi retencję.

Koszt centralnego logowania rośnie w trzech wymiarach naraz: ilością logowanych zdarzeń (wolumen wejściowy), długością retencji (jak długo trzymamy indeks) i liczbą pól indeksowanych z każdego zdarzenia (bardziej szczegółowe indeksowanie zajmuje więcej miejsca i więcej czasu procesora na zapis). Świadomy kompromis — nie każdy log aplikacyjny wymaga tej samej retencji co logi audytowe systemu uwierzytelniania — jest tańszy niż jedna uniwersalna, maksymalnie długa retencja dla wszystkiego.

Częste błędy

Wysyłanie wszystkiego z maksymalną retencją bez rozróżnienia. Log debugowania aplikacji generowany tysiąc razy na sekundę i log audytowy logowań do systemu produkcyjnego nie zasługują na tę samą retencję ani ten sam priorytet przechowywania — brak tego rozróżnienia kończy się albo astronomicznym kosztem magazynu, albo skróceniem retencji dla wszystkiego, łącznie z danymi, których faktycznie potrzebujemy trzymać długo.

Poleganie wyłącznie na logu lokalnym „bo przecież mamy monitoring". Monitoring (temat rozdziału Architektura Zabbixa: serwer, proxy, agent, frontend) mówi, że coś jest nie tak — centralne logowanie mówi dlaczego. To są dwa uzupełniające się, ale różne narzędzia, i żadne z nich nie zastępuje drugiego.

Brak testu end-to-end po skonfigurowaniu wysyłki. Skonfigurowanie transportu bez wysłania i zweryfikowania testowego zdarzenia zostawia otwartą możliwość, że logi po cichu nigdzie nie docierają — od miesiąca życzeniowego przekonania, że coś jest logowane, do odkrycia pustki dokładnie w chwili, w której log jest potrzebny do wyjaśnienia incydentu.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — dobór retencji

Trzy typy zdarzeń: logi debugowania aplikacji testowej, logi uwierzytelniania SSH na serwerach produkcyjnych, logi żądań HTTP z reverse proxy. Proponujemy dla każdego orientacyjną retencję i uzasadniamy różnicę.

Odpowiedź

Logi debugowania środowiska testowego — krótka retencja (dni), bo mają wartość wyłącznie w trakcie aktywnego debugowania i generują duży wolumen o niskiej wartości długoterminowej. Logi uwierzytelniania SSH na produkcji — długa retencja (miesiące, często wymuszona wymaganiami audytowymi), bo są dowodem w razie incydentu bezpieczeństwa wykrytego z opóźnieniem. Logi żądań HTTP z proxy — retencja pośrednia (tygodnie), wystarczająca do analizy wzorców ruchu i diagnozowania problemów wydajnościowych bez trzymania ich w nieskończoność.

Pytania kontrolne

1. Dlaczego kierunek ruchu sieciowego przy centralnym logowaniu jest zawsze od hosta źródłowego do serwera, a nie odwrotnie?

Odpowiedź

Bo centralne logowanie działa w modelu push — to źródło samo, z własnej inicjatywy, wysyła zdarzenie w chwili jego wystąpienia, zamiast czekać na zapytanie centralnego systemu. Firewall na hoście źródłowym musi więc przepuszczać ruch wychodzący do serwera logowania; sam serwer logowania nie musi mieć żadnej trasy inicjującej połączenie do monitorowanych hostów.

2. Dlaczego log ustrukturyzowany (np. GELF) bywa tańszy w utrzymaniu niż nieustrukturyzowany, mimo że wymaga więcej pracy przy pisaniu aplikacji?

Odpowiedź

Bo podział na pola jest ustalony raz, w kodzie aplikacji, i nie zależy od reguły parsowania utrzymywanej osobno po stronie centralnego systemu logowania — zmiana formatu komunikatu tekstowego w logu nieustrukturyzowanym wymaga aktualizacji reguły ekstrakcji pól za każdym razem, gdy format się zmieni, co jest źródłem cichych awarii wyszukiwania opisanych przy rozszerzeniach Graylogu.

Powiązane: Graylog: wejścia, strumienie, wyszukiwanie · Wysyłka logów z hostów · Retencja logów

Źródła: dokumentacja Graylog, rozdział „Getting Started" · RFC 5424 (The Syslog Protocol)

Graylog: wejścia, strumienie, wyszukiwanie

Poziom: średnio **Dotyczy: Graylog 7.1, OpenSearch 2.19, MongoDB 8.0**

Czas: ~30 min

Graylog sam nie przechowuje ani nie przeszukuje logów — to warstwa nad dwoma innymi systemami, więc zaczynamy od tego, co robi każdy z trzech elementów, zanim przejdziemy do konfiguracji wejść.

Skróty użyte na tej stronie (6)

Skrót	Co znaczy
GELF	Graylog Extended Log Format: format logów Grayloga
HTTP	Hypertext Transfer Protocol: protokół stron WWW
JSON	JavaScript Object Notation: tekstowy format danych z klamrami i cudzysłowami
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia
UDP	User Datagram Protocol: szybki protokół sieciowy bez potwierdzeń
VLAN	Virtual LAN: wydzielona logicznie sieć na tym samym sprzęcie

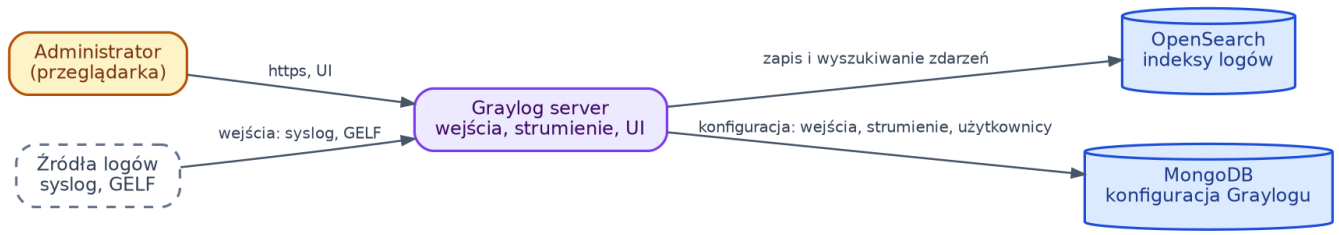
Z życia wzięte

Z ŻYCIA WZIĘTE

W piątek po południu interfejs Graylogu przestaje pokazywać nowe zdarzenia, mimo że aplikacje na wszystkich hostach dalej normalnie działają. Tomek sprawdza najpierw MongoDB, bo tam trzyma się konfiguracja, i widzi, że proces działa bez zarzutu. Dopiero w logu samego Graylogu znajduje błędy połączenia z OpenSearch, gdzie skończyło się miejsce na dysku. Po zwolnieniu miejsca i restarcie usługi napływ nowych logów wraca, a stare dane w OpenSearch okazują się nietknięte przez cały czas przerwy.

Architektura — trzy elementy, trzy różne role

> **Pojęcie: Graylog (server).** Warstwa aplikacyjna przyjmująca logi na skonfigurowanych wejściach, przepuszczająca je przez rozszerzenia i strumienie, i wystawiająca interfejs webowy do wyszukiwania, budowania pulpitów i definiowania alertów. Sam Graylog nie trzyma danych trwale — to proces pośredniczący między transportem a silnikiem wyszukiwania.



Graylog przyjmuje logi ze źródeł, zapisuje i przeszukuje je w OpenSearch, a swoją konfigurację trzyma w MongoDB (kliknięcie otwiera pełny rozmiar)

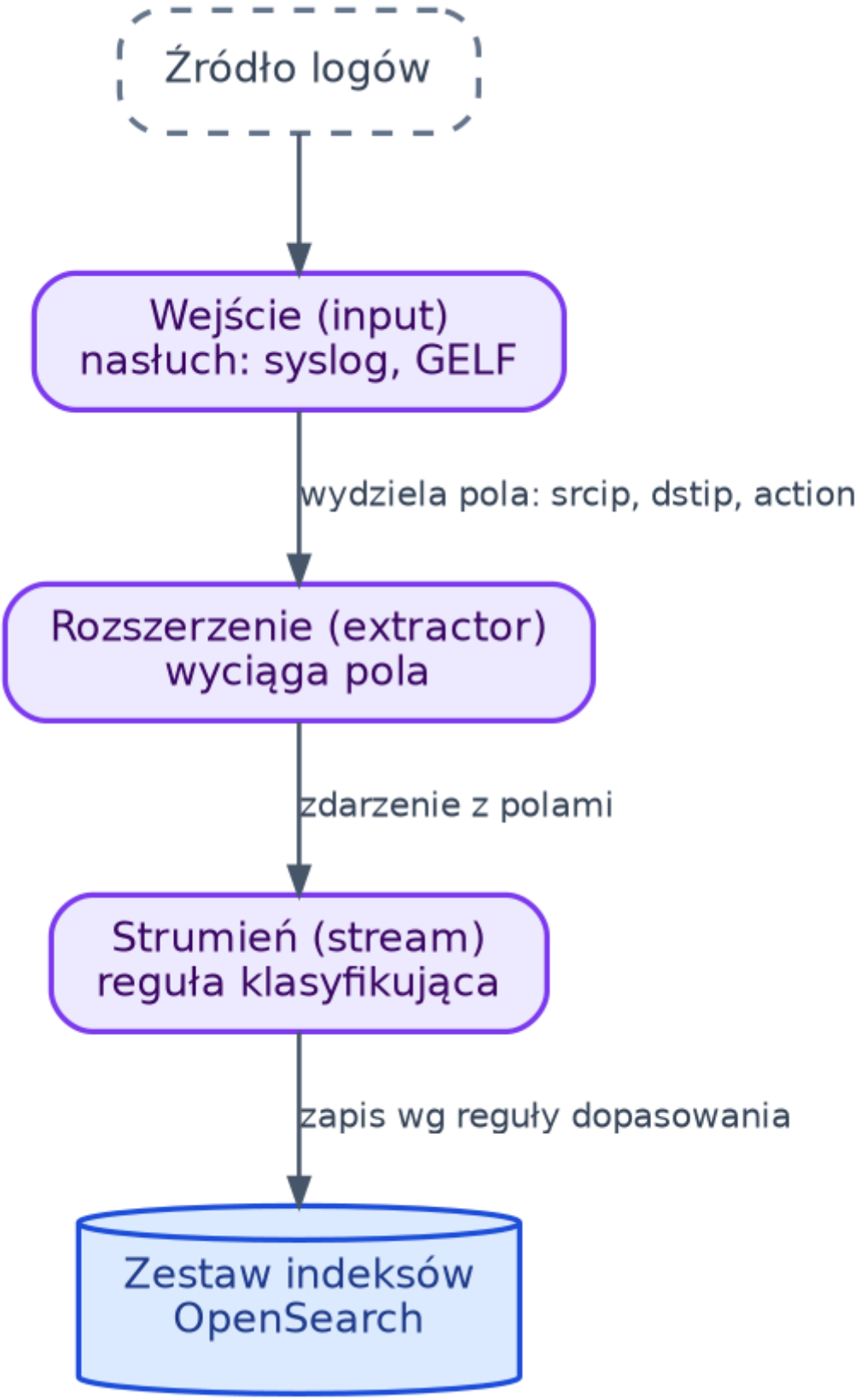
> **Pojęcie: OpenSearch (dawniej Elasticsearch).** Silnik wyszukiwania pełnotekstowego, w którym faktycznie leżą zindeksowane logi. Graylog komunikuje się z nim przy każdym zapytaniu wyszukiwania i przy każdym zapisie nowego zdarzenia — utrata OpenSearch oznacza utratę samych danych, nie tylko interfejsu do ich przeglądania.

> **Pojęcie: MongoDB.** Baza trzymająca konfigurację samego Graylogu — definicje wejść, strumieni, użytkowników, zapisanych wyszukiwań i pulpitów. MongoDB nie przechowuje treści logów — jej utrata oznacza utratę konfiguracji systemu, przy zachowanych, wciąż leżących w OpenSearch, danych.

Rozdzielenie tych trzech ról ma bezpośrednią konsekwencję operacyjną: kopię zapasową samej konfiguracji Graylogu (MongoDB) i kopię zapasową danych logów (indeksy OpenSearch) planujemy i wykonujemy jako dwie osobne procedury, o zupełnie różnym rozmiarze i częstości.

Wejście — punkt wejścia danych

Pojęcie: wejście (input). Nasłuch skonfigurowany w Graylogu na określonym porcie i protokole, przyjmujący logi z zewnątrz. Jeden serwer Graylog obsługuje jednocześnie wiele wejść różnych typów — nic nie stoi na przeszkodzie, żeby ten sam serwer przyjmował syslog UDP na porcie 1514 i GELF TCP na porcie 12201 równolegle.



Zdarzenie płynie od źródła przez wejście, przez rozszerzenie wydzielające pola, do strumienia, który kieruje je do właściwego zestawu indeksów (kliknięcie otwiera pełny rozmiar)

Syslog UDP — najprostszy do skonfigurowania, ale bez potwierdzenia dostarczenia; komunikat zgubiony w sieci znika bez śladu i bez błędu po żadnej ze stron. Stosujemy go tam, gdzie źródło (typowo urządzenie sieciowe) nie wspiera nic innego.

Syslog TCP — to samo co powyżej, ale z gwarancją transportową TCP; komunikat albo dotrze, albo nadawca dostanie błąd połączenia. Preferowany wariant, gdy źródło wspiera oba.

GELF UDP/TCP/HTTP — format natywny dla Graylogu, niosący ustrukturyzowane pola JSON zamiast płaskiego tekstu syslog. Aplikacje wysyłające logi bezpośrednio (przez bibliotekę GELF) omijają w ten sposób etap parsowania nieustrukturyzowanego tekstu.

Rozszerzenia (extractors)

> **Pojęcie: rozszerzenie (extractor).** Reguła przypisana do konkretnego wejścia, wyciągająca pojedyncze pola ze surowej treści przychodzącego komunikatu — wyrażeniem regularnym, podziałem po separatorze, albo gotowym wzorcem (np. Grok) — zanim zdarzenie trafi dalej do strumieni i indeksu. Komunikat syslog z urządzenia sieciowego przychodzi jako jeden ciąg tekstu; dopiero rozszerzenie wydziela z niego pola typu `srcip`, `dstip` czy `action`, które potem stają się przeszukiwalne osobno, a nie tylko jako fragment większego tekstu.

Uważaj: rozszerzenia działają w kolejności zdefiniowanej na wejściu i każde kolejne pracuje na wyniku poprzedniego — źle poukładana kolejność (np. rozszerzenie zależne od pola, które dopiero następne w kolejności wydzieli) po cichu nie wydzieli niczego, bez żadnego błędu widocznego od razu.

Strumienie i reguły kierowania

> **Pojęcie: strumień (stream).** Reguła klasyfikująca przychodzące zdarzenia w czasie rzeczywistym i kierująca dopasowane do osobnego, nazwanego kontenera — logi firewalla do jednego strumienia, logi aplikacji webowej do drugiego, niezależnie od tego, którym wejściem fizycznie dotarły. Strumień jest podstawową jednostką, na której budujemy zapisane wyszukiwania, pulpity i — co najważniejsze operacyjnie — alerty.

> **Pojęcie: reguła strumienia.** Warunek dopasowania oparty o wartość konkretnego pola (np. `facility` równe `local0`, albo `source` zaczynające się od `fw-`) — zdarzenie trafia do

strumienia, jeśli spełnia wszystkie (albo dowolny, zależnie od ustawienia) skonfigurowane warunki.

Indeks i zestaw indeksów

Pojęcie: zestaw indeksów (index set). Grupa fizycznych indeksów OpenSearch przypisana do jednego lub kilku strumieni, z własną, niezależną konfiguracją rotacji i retencji. Rozdzielenie zestawów indeksów pozwala trzymać logi audytowe miesiącami, a logi debugowania aplikacji testowej dniami — dokładnie ten kompromis retencji, o którym mowa na stronie Po co centralne logowanie i jak to działa, realizuje się technicznie właśnie przez osobne zestawy indeksów per strumień.

Składnia wyszukiwania

Słownik podstawowych pól i operatorów, z jakich w praktyce najczęściej korzystamy przy analizie logów firewalla i ruchu sieciowego:

Pole / operator	Znaczenie
`srcip`	adres IP źródłowy zdarzenia
`dstip`	adres IP docelowy
`dstport`	port docelowy
`remote_addr`	adres IP klienta wykonującego żądanie (typowo z logów Nginksa)
`request_path`	fragment adresu URL od pierwszego `/`
`action`	akcja podjęta przez firewall lub urządzenie — np. `deny`, `accept`
`policyid`	identyfikator reguły (polityki) firewalla, która obsłużyła ruch
`AND`	koniunkcja — oba warunki muszą być spełnione
`OR`	alternatywa — wystarczy spełnienie jednego z warunków
`AND NOT`	wykluczenie danego warunku z wyniku

Przykład zapytania łączącego kilka pól:

```
action:deny AND dstport:22 AND NOT srcip:10.20.0.0/16
```

Wyszukujemy wszystkie odrzucone przez firewall próby połączenia na port 22, pochodzące spoza naszej sieci wewnętrznej (VLAN serwerów `10.20.10.0/24`, biura `10.20.20.0/24` i pozostałe zawierają się w `10.20.0.0/16`) — typowy pierwszy krok przy podejrzeniu skanowania portów z zewnątrz.

Zapisane wyszukiwania i pulpity

> **Pojęcie: zapisane wyszukiwanie.** Zapytanie wraz z zakresem czasowym i wybranymi polami, zapamiętane pod nazwą do ponownego uruchomienia bez odtwarzania składni za każdym razem — powtarzalne pytania („ile odrzuceń z danej podsieci w ostatniej dobie") zamieniamy w zapisane wyszukiwanie zamiast wklejać ten sam ciąg tekstu co tydzień.

> **Pojęcie: pulpit (dashboard).** Zestaw widgetów (wykresów, liczników, tabel) zbudowanych na wynikach zapisanych wyszukiwań, prezentowany na jednym ekranie — odpowiednik pulpitu Zabbixa, ale zbudowany na wynikach wyszukiwania tekstowego, a nie na metrykach liczbowych.

Alerty i powiadomienia

> **Pojęcie: definicja alertu.** Reguła uruchamiana cyklicznie nad danym strumieniem, sprawdzająca warunek (np. „więcej niż 50 zdarzeń `action:deny` w ciągu 5 minut z tego samego `srcip`") i wyzwalająca powiadomienie, gdy warunek zostanie spełniony. Mechanizm koncepcyjnie odpowiada wyzwalaczowi Zabbixa opisanemu na stronie Host, element danych, wyzwalacz, akcja: jak to się składa — różnica leży w tym, że warunek Graylogowego alertu operuje na wynikach wyszukiwania tekstowego, a nie na wartości liczbowej pojedynczego elementu danych.

> **Pojęcie: powiadomienie (notification).** Kanał, którym wyzwolony alert dociera do odbiorcy — e-mail, webhook do zewnętrznego systemu zgłoszeń, albo integracja ze Slackiem/Teams. Jedna definicja alertu może mieć podpiętych kilka powiadomień naraz.

Częste błędy

Brak strumienia dedykowanego dla danych krytycznych. Logi audytowe wpadające do domyślnego strumienia razem z całym pozostałym ruchem dzielą z nim tę samą retencję i nie pozwalają zbudować osobnego, celowanego alertu bez przefiltrowywania szumu za każdym razem.

Rozszerzenie źle dopasowane do formatu, który się zmienił. Aktualizacja aplikacji albo urządzenia sieciowego zmieniająca format logu po cichu psuje istniejące rozszerzenie — pola przestają się wydzielać, zapytania po tych polach zaczynają zwracać puste wyniki, a przyczyna nie jest widoczna bez porównania surowej treści komunikatu sprzed i po zmianie.

Alert zbudowany na zbyt szerokim warunku. Definicja alertu bez rozsądnego progu czasowego i ilościowego (np. „dowolne wystąpienie `action:deny`" zamiast „więcej niż 50 w 5 minut") zalewa odbiorców powiadomieniami do tego stopnia, że przestają je czytać — dokładnie ten sam mechanizm zmęczenia alertami, co przy źle dobranych progach w Zabbiksie.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — budowa zapytania

Budujemy zapytanie wyszukujące wszystkie żądania HTTP do ścieżek zaczynających się od `/admin`, z wyłączeniem żądań pochodzących z sieci biurowej `10.20.20.0/24`.

Odpowiedź

```
request_path:\/admin* AND NOT remote_addr:10.20.20.0/24
```

Gwiazdka na końcu dopasowuje dowolne dokończenie ścieżki, ukośnik poprzedzamy odwrotnym ukośnikiem, bo w składni Lucene gołe `/` otwiera wyrażenie regularne, a `AND NOT` wyklucza ruch z sieci biurowej.

Uważaj: nie zaczynamy wzorca od gwiazdki. Zapytanie `request_path:*admin*` ma wiodący wildcard, a wiodące wildcardy są w Graylogu domyślnie wyłączone (`allow_leading_wildcard_searches = false`), więc zwróci błąd zamiast wyniku. W kontenerze włącza je zmienna `GRAYLOG_ALLOW_LEADING_WILDCARD_SEARCHES=true`, ale takie zapytania przeglądają cały indeks i mocno obciążają OpenSearch, więc w Zielonej zostawiamy wyłączone.

Szczegóły są na stronie Graylog: codzienna praca z logami.

Pytania kontrolne

1. Dlaczego utrata bazy MongoDB i utrata indeksów OpenSearch to dwa różne rodzaje awarii?

Odpowiedź

MongoDB trzyma konfigurację Graylogu — wejścia, strumienie, użytkowników, definicje alertów — bez niej system przestaje wiedzieć, jak klasyfikować i prezentować dane, ale same logi wciąż istnieją w OpenSearch. Utrata OpenSearch to utrata samej treści logów — nawet z zachowaną konfiguracją nie ma czego przeszukiwać. Odtworzenie po pierwszej awarii to odtworzenie konfiguracji od zera lub z kopii; po drugiej — nieodwracalna utrata historii zdarzeń sprzed awarii.

Powiązane: Po co centralne logowanie i jak to działa · Wysyłka logów z hostów · Host, element danych, wyzwalacz, akcja: jak to się składa

Źródła: dokumentacja Graylog, rozdział „Inputs", „Streams" i „Alerts"