Wysyłka logów

Konfiguracja rsyslog i journald.

- Wysyłka logów z hostów

Wysyłka logów z hostów

Poziom: średnio

Dotyczy: rsyslog (RHEL/Rocky, Debian/Ubuntu), systemd-journald, Windows Server

Czas: ~25 min

Skonfigurowanie centralnego systemu logowania nie ma sensu, dopóki żaden host faktycznie do niego nic nie wysyła — tę stronę traktujemy jako drugą połowę pary z Graylog: wejścia, strumienie, wyszukiwanie.

> W przykładach wysyłamy na port `1514`, a nie na klasyczny `514`. Graylog w kontenerze działa jako zwykły użytkownik i nie może nasłuchiwać na porcie poniżej 1024, dlatego wejście syslog w Graylogu ustawiamy na `1514`, a ten sam port publikuje `compose.yaml` z instalacji (Instalacja Grayloga w Dockerze: Graylog, MongoDB i OpenSearch). Gdy jakieś urządzenie potrafi wysyłać tylko na `514` (część sprzętu sieciowego), robimy przekierowanie na hoście Dockera albo zmieniamy mapowanie portów na `"514:1514/udp"` i `"514:1514/tcp"`.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
GELF	Graylog Extended Log Format: format logów Grayloga
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
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
TCP	Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia
UDP	User Datagram Protocol: szybki protokół sieciowy bez potwierdzeń
WAN	Wide Area Network: sieć rozległa, czyli strona zapory zwrócona do internetu, na przykład interfejsy wan1 i wan2 zapory fw01

Fiszka

TYLKO ODCZYT

Wysyłka przez TCP: składnia @@host:port

```
*.* @@logi.zielona.example:1514
```

Podwójny znak `@@` oznacza transport TCP — z gwarancją dostarczenia w ramach pojedynczego połączenia, ale bez retransmisji po jego zerwaniu bez dodatkowej konfiguracji kolejki, opisanej niżej.

TCP = Transmission Control Protocol: protokół sieciowy z potwierdzaniem dostarczenia

TYLKO ODCZYT

Wysyłka przez UDP: składnia @host:port

```
*.* @logi.zielona.example:1514
```

Pojedynczy `@` oznacza transport UDP — najprostszy, ale bez żadnej gwarancji dostarczenia; komunikat zgubiony po drodze nie generuje żadnego błędu po stronie nadawcy.

UDP = User Datagram Protocol: szybki protokół sieciowy bez potwierdzeń

TYLKO ODCZYT

Filtrowanie: wysyłka auth/authpriv od poziomu warning

```
auth,authpriv.warning @@logi.zielona.example:1514
```

Wysyłka wyłącznie komunikatów uwierzytelniania od poziomu ostrzeżenia w górę.

TYLKO ODCZYT

Filtrowanie: wysyłka wszystkiego poza local0.debug

```
*.*;local0.!=debug @@logi.zielona.example:1514
```

Wysyłka wszystkiego oprócz kategorii debugowania aplikacji lokalnej. Wykrzyknik przed nazwą severity odwraca dopasowanie — powyższa reguła przekazuje wszystko, ale z kategorii `local0` pomija konkretnie poziom `debug`.

PLIK KONFIGURACYJNY

Szablon formatowania komunikatu: template GraylogFormat

```
template(name="GraylogFormat" type="string"
  string="<%PRI%>1 %TIMESTAMP:::date-rfc3339% %HOSTNAME% %APP-NAME% %PROCID% %MSGID% -
%MSG%\n"
)


*.* @@logi.zielona.example:1514;GraylogFormat
```

Powyższy szablon składa komunikat w formacie syslog RFC 5424 (nowszy, ustrukturyzowany wariant standardu, w odróżnieniu od starszego RFC 3164) — Graylog rozpoznaje oba, ale RFC 5424 niesie więcej ustrukturyzowanych pól bez potrzeby dodatkowego rozszerzenia po stronie wejścia.

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

PLIK KONFIGURACYJNY

Kolejkowanie akcji na wypadek niedostępności odbiorcy: action omfwd

```
action(type="omfwd"
  target="logi.zielona.example" port="1514" protocol="tcp"
  queue.type="linkedList"
  queue.filename="fwdRule1"
  queue.maxDiskSpace="1g"
  queue.saveOnShutdown="on"
  action.resumeRetryCount="-1"
)
```

TYLKO ODCZYT

Sprawdzenie składni pliku konfiguracyjnego: rsyslogd -N1

```
sudo rsyslogd -N1 -f /etc/rsyslog.conf
```

Sprawdzamy składnię pliku konfiguracyjnego przed przeładowaniem — analogicznie do `nginx -t` w konfiguracji Nginksa.

Przykład:

```
rsyslogd: End of config validation run. Bye.
```

ZMIENIA SYSTEM

Przeładowanie usługi rsyslog: systemctl restart

```
sudo systemctl restart rsyslog
```

Dopiero po pozytywnym teście przeładowujemy usługę.

systemctl = system control: sterowanie usługami systemd

PLIK KONFIGURACYJNY

Włączenie przekazywania journald do syslog: ForwardToSyslog

```
[Journal]
ForwardToSyslog=yes
```

ZMIENIA SYSTEM

Restart usługi systemd-journald po zmianie: systemctl restart

```
sudo systemctl restart systemd-journald
```

Po zmianie w `/etc/systemd/journald.conf` restartujemy usługę.

Uważaj: samo `ForwardToSyslog=yes` przekazuje komunikaty z `journald` do lokalnego `rsyslog` — to `rsyslog`, skonfigurowany regułą z sekcji wyżej, odpowiada dopiero za faktyczną wysyłkę na zewnątrz. Bez działającej reguły przekazania w `rsyslog.conf` logi zatrzymają się lokalnie, mimo poprawnie ustawionego `journald`.

systemctl = system control: sterowanie usługami systemd

□□ TYLKO ODCZYT

Test wysyłki pojedynczego komunikatu: logger

```
logger -p local0.info "test wysylki do centralnego logowania $(hostname)"
```

Generujemy jeden, łatwo odróżnialny od reszty ruchu komunikat testowy, z nazwą hosta w treści, żeby jednoznacznie odnaleźć go po drugiej stronie.

□□ **logger** = Logger: zapisuje komunikat w dzienniku systemowym

□□ TYLKO ODCZYT

Sprawdzenie po stronie odbiorcy: nasłuch tcpdump

```
sudo tcpdump -n -i any port 1514
```

Wykonujemy na serwerze centralnym, zanim jeszcze sprawdzimy interfejs Graylogu — potwierdza, że pakiety w ogóle docierają na poziomie sieci, niezależnie od tego, czy wejście Graylogu jest poprawnie skonfigurowane.

Przykład:

```
14:02:11.884213 IP app01.ipa.zielona.example.51204 > logi.zielona.example.1514: UDP,
length 98
```

Dopiero po potwierdzeniu ruchu na poziomie sieci szukamy komunikatu testowego w interfejsie Graylogu, filtrując po treści wpisanej komendą `logger`.

□□ **tcpdump** = podsłuch ruchu sieciowego na interfejsie

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

O 8:40 Tomek dostaje pytanie, dlaczego logi uwierzytelniania z `app01` urwały się na dwie godziny w nocy, akurat gdy łącze do centrum danych miało krótką awarię. Sprawdza konfigurację rsyslog na tym hoście i widzi, że reguła wysyłki nie ma skonfigurowanej kolejki dyskowej, więc komunikaty z okna niedostępności `logi.zielona.example` po prostu przepadły bez śladu. Dokłada `queue.type`, `queue.filename` i `action.resumeRetryCount=-1`, tak żeby

kolejna, podobna przerwa w łączu skończyła się buforowaniem, a nie cichą utratą danych.

Konfiguracja rsyslog do wysyłki na serwer centralny

Pojęcie: rsyslog. Demon logowania systemowego, domyślny w większości dystrybucji Linuksa, odpowiedzialny za odbiór komunikatów z jądra, `journald` i aplikacji, ich lokalny zapis do `/var/log`, oraz — istotne tutaj — ich przekazanie dalej do zdalnego odbiorcy.

Pojęcie: selektor `*.*`. Pierwszy człon reguły rsyslog, złożony z pary facility.severity, gdzie gwiazdka oznacza „dowolna wartość" — `*.*` przekazuje dalej wszystkie komunikaty niezależnie od źródła i poziomu ważności, co w praktyce stosujemy rzadko, wolimy filtrować już na tym etapie.

Filtrowanie po facility i severity

Pojęcie: facility. Kategoria źródła komunikatu w standardzie syslog — `auth`/`authpriv` (uwierzytelnianie), `daemon` (usługi systemowe), `kern` (jądro), `local0`–`local7` (kategorie zarezerwowane do własnego użytku, typowo przypisywane konkretnej aplikacji).

Pojęcie: severity. Poziom ważności komunikatu, od `debug` (najniższy) przez `info`, `notice`, `warning`, `err`, `crit`, `alert`, po `emerg` (najwyższy).

Szablony formatowania komunikatu

Pojęcie: szablon (template). Definicja formatu, w jakim rsyslog składa wysyłany komunikat — przydatna, gdy chcemy dodać do standardowego komunikatu syslog własne pole (np. nazwę środowiska) albo dopasować format do oczekiwań konkretnego wejścia po stronie Graylogu.

Kolejkowanie na wypadek niedostępności odbiorcy

Pojęcie: kolejka akcji (action queue). Bufor rsyslog przechowujący komunikaty, których nie udało się jeszcze dostarczyć do zdalnego odbiorcy — bez tego mechanizmu chwilowa niedostępność serwera centralnego (restart, aktualizacja, krótka awaria sieci) oznacza trwałą utratę logów wygenerowanych w tym okienku.



Rsyslog buforuje komunikaty w kolejce na dysku, gdy serwer centralny jest niedostępny, i wznawia wysyłkę po powrocie połączenia (kliknięcie otwiera pełny rozmiar)

Pojęcie: queue.saveOnShutdown. Zapisuje zawartość kolejki na dysk przy zatrzymaniu usługi, zamiast trzymać ją wyłącznie w pamięci — restart usługi rsyslog (np. przy aktualizacji pakietu) nie gubi wtedy komunikatów oczekujących na wysyłkę.

Pojęcie: action.resumeRetryCount. Liczba prób ponowienia wysyłki po niepowodzeniu, gdzie `-1` oznacza próbowanie bez ograniczenia, aż odbiorca wróci — w połączeniu z kolejką zapisaną na dysku daje to odporność na dłuższe, wielogodzinne przerwy w łączności z serwerem centralnym.

journald i ForwardToSyslog

> **Pojęcie: journald.** Domyślny na systemach z systemd dziennik zdarzeń binarny, opisany szerzej przy retencji na stronie Retencja logów — na maszynach, gdzie to `journald`, a nie `rsyslog`, jest głównym punktem zbierania logów systemowych, przekazanie ich dalej do centralnego serwera wymaga jawnego włączenia przekazywania do syslog.

Wysyłka z Windowsa w skrócie

> **Pojęcie: NXLog.** Agent logowania dla Windows, funkcjonalnie odpowiadający `rsyslog` po stronie Linuksa — czyta dziennik zdarzeń Windows (Event Log) i przekazuje go dalej jako syslog albo GELF, z własną konfiguracją filtrowania po kanale i poziomie zdarzenia.
>
> **Pojęcie: Winlogbeat.** Alternatywny agent z rodziny Beats, wysyłający zdarzenia z dziennika Windows bezpośrednio do OpenSearch/Elasticsearch albo przez Logstash, z gotowymi modułami do typowych źródeł (zabezpieczenia, Sysmon, PowerShell).

Oba narzędzia rozwiązują ten sam problem — dziennik zdarzeń Windows nie jest plikiem tekstowym w rozumieniu `rsyslog`, więc wymaga dedykowanego agenta potrafiącego czytać go przez API Windows, zamiast prostego `tail` na pliku.

Częste błędy

Reguła wysyłki umieszczona przed regułami lokalnego zapisu w pliku konfiguracyjnym. Kolejność reguł w `rsyslog.conf` ma znaczenie — reguła wysyłki umieszczona zbyt wcześnie, przed regułami filtrującymi hałas, wysyła na zewnątrz więcej, niż faktycznie zamierzaliśmy.

Brak kolejki dyskowej przy transporcie TCP. Sam TCP gwarantuje dostarczenie w ramach istniejącego połączenia, ale nie odtwarza automatycznie utraconych komunikatów po zerwaniu i ponownym nawiązaniu połączenia — bez skonfigurowanej kolejki (`queue.type`, `queue.filename`) komunikaty z okna niedostępności odbiorcy przepadają bezpowrotnie.

Testowanie wyłącznie przez odczekanie i sprawdzenie interfejsu. Bez kroku pośredniego (`tcpdump` na porcie odbiorczym) nie da się jednoznacznie odróżnić „logi nie dotarły przez sieć" od „logi dotarły, ale wejście Graylogu jest źle skonfigurowane" — oba objawiają się identycznie jako brak wyniku w wyszukiwarce.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — dobór transportu

Serwer wysyła logi audytowe uwierzytelniania do centralnego systemu logowania przez sieć rozległą (WAN) o niestabilnym łączu. Uzasadniamy wybór między UDP a TCP wraz z kolejkowaniem.

Odpowiedź

TCP z kolejką dyskową (`queue.type`, `queue.saveOnShutdown`) i nieograniczoną liczbą prób ponowienia — logi audytowe uwierzytelniania mają wartość dowodową, więc utrata komunikatów przy niestabilnym łączu, jaką dopuszcza UDP bez żadnego potwierdzenia, jest tu nieakceptowalna. Koszt w postaci nieco większego obciążenia połączenia jest tu drugorzędny wobec ryzyka cichej utraty danych.

Powiązane: Po co centralne logowanie i jak to działa · Graylog: wejścia, strumienie, wyszukiwanie · Retencja logów

Źródła: `man 5 rsyslog.conf` · dokumentacja rsyslog, rozdział „Reliable Forwarding" · RFC 5424 · dokumentacja Graylog, rozdział „Sending logs"