rsyslog

Если сообщения пересылаются между хостами, использующими rsyslog, можно вместо plain TCP sysog использовать RELP — Reliable Event Logging Protocol. Был создан для rsyslog, сейчас поддерживается и некоторыми другими системами. В частности, его понимают Logstash и Graylog. Для транспорта использует TCP. Может опционально шифровать сообщения с помощью TLS. Надёжнее plain TCP syslog, не теряет сообщения при разрыве соединения. Решает проблему с многострочными сообщениями.

Конфигурация rsyslog

В отличии от второй распространённой альтернативы, syslog-ng, rsyslog совместим с конфигами исторического syslogd:

```
auth,authpriv.*            /var/log/auth.log
*.*;auth,authpriv.none     /var/log/syslog
*.*       @syslog.example.net
```

Т. к. возможности rsyslog гораздо больше, чем у его предшественника, формат конфигов был расширен дополнительными директивами, начинающимися со знака $:

```
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$WorkDirectory /var/spool/rsyslog
$IncludeConfig /etc/rsyslog.d/*.conf
```

Начиная с шестой версии, появился си-подобный формат RainerScript, позволяющий задавать сложные правила обработки сообщений.

Т. к. всё это делалось постепенно и с учётом совместимости со старыми конфигами, то в итоге получилась пара неприятных моментов:

- некоторые плагины(я пока с такими не сталкивался) могут не поддерживать новый RainerScript стиль настроек, им по-прежнему нужны старые директивы

- настройка через старые директивы не всегда работает как ожидается для нового формата:
    - если модуль omfile вызывается с помощью старого формата:
      auth,authpriv.* /var/log/auth.log, то владелец и разрешения получившегося файла регулируются старыми
      директивами $FileOwner, $FileGroup, $FileCreateMode. А вот если он вызывается с помощью action(type="omfile" ...), то эти директивы игнорируются, и надо настраивать параметы action или задавать при загрузке модуля
    - Директивы вида $ActionQueueXXX настраивают только ту очередь, которая будет использована в первом action после них, потом значения сбрасываются.
- точки с запятой где-то запрещены, а где-то наоборот обязательны(второе реже)

Чтобы не спотыкаться об эти тонкости(да, в документации они описаны, но кто же её целиком читает?), стоит следовать простым правилам:

- для маленьких простых конфигов использовать старый формат:
  :programname, startswith, "haproxy" /var/log/haproxy.log
- для сложной обработки сообщений и для тонкой настройки Actions всегда использовать RainerScript, не трогая legacy директивы вида $DoSomething

Подробнее про формат конфига здесь.

Обработка сообщений

- Все сообщения приходят из Input(их может быть много) и попадают на обработку в привязанный к нему RuleSet. Если это явно не задано, то сообщения попадут в RuleSet по-умолчанию. Все директивы обработки сообщений, не вынесенные в отдельные RuleSet-блоки, относятся именно к

нему. В частности, к нему относятся все директивы из традиционного формата конфигов:
local7.* /var/log/myapp/my.log

- к Input привязан список парсеров для разбора сообщения. Если явно не задано, будет использоваться список парсеров для разбора традиционного формата syslog
- Парсер выделяет из сообщения свойства. Самые используемые:
    - $msg – сообщение
    - $rawmsg – сообщение целиком до обработки парсером
    - $fromhost, $fromhost-ip – DNS имя и IP адрес хоста-отправителя
    - $syslogfacility, $syslogfacility-text – facility в числовой и текстовой форме
    - $syslogseverity, $syslogseverity-text – то же для severity
    - $timereported – время из сообщения
    - $syslogtag – поле TAG
    - $programname – поле TAG с отрезанным id процесса: named[12345] -> named
    - весь список можно посмотреть тут
- RuleSet содержит список правил, правило состоит из фильтра и привязанных к нему одного или нескольких Actions
- Фильтры – логические выражения, с использованием свойств сообщения. Подробнее про фильтры
- Правила применяются последовательно к сообщению, попавшему в RuleSet, на первом сработавшем правиле сообщение не останавливается
- Чтобы остановить обработку сообщения, можно использовать специальное discard action: stop или ~ в легаси-формате.
- Внутри Action часто используются шаблоны. Шаблоны позволяют генерировать данные для передачи в Action из свойств сообщения, например, формат сообщения для передачи по сети или имя файла для записи. Подробнее про шаблоны
- Как правило, Action использует модуль вывода(“om…”) или

модуль изменения сообщения(“mm…”). Вот некоторые из них:

- omfile – вывод в файл
- omfwd – пересылка по сети, через udp или tcp
- omrelp – пересылка по сети по протоколу RELP
- onmysql, ompgsql, omoracle – запись в базу
- omelasticsearch – запись в ElasticSearch
- omamqp1 – пересылка по протоколу AMQP 1.0
- весь список модулей вывода

Подробнее про порядок обработки сообщений

Примеры конфигурации

Записываем все сообщения категорий auth и authpriv в файл /var/log/auth.log, и продолжаем их обработку:

```
# legacy
auth,authpriv.*  /var/log/auth.log
# новый формат
if ( $syslogfacility-text == "auth" or $syslogfacility-text ==
"authpriv" ) then {
    action(type="omfile" file="/var/log/auth.log")
}
```

Все сообщения с именем программы, начинающимся с “haproxy”, записываем в файл /var/log/haproxy.log, не сбрасывая буфер на диск после записи каждого сообщения, и прекращаем дальнейшую обработку:

```
# legacy (обратите внимание на минус перед именем файла,
отключающий сброс буфера)
:programname, startswith, "haproxy", -/var/log/haproxy.log
& ~
# новый формат
if ( $programname startswith "haproxy" ) then {
       action(type="omfile"  file="/var/log/haproxy.log"
flushOnTXEnd="off")
    stop
}
# можно совмещать
if   $programname   startswith   "haproxy"   then   -
```

/var/log/haproxy.log
&~

Проверка конфига: rsyslogd -N 1. Больше примеров конфигурации: site.

Источник