Syslog and CEF Advanced Filtering at the Rsyslog Server

Last updated by | Naser Mohammed | Mar 19, 2023 at 7:18 AM CDT

Contents

Azure Sentinel

I. Syslog Ingestion Guide -

Purpose:

The goal of this doc is to document and outline a basic strategy for Syslog message ingestion, most especially to capture the somewhat painful learning experiences.

II. High Level - System Architecture:

Syslog clients are configured to send messages to the Syslog Server whereupon the Syslog Server will just bounce the events over to the local OMS Agent, there is no need to store the remote messages locally. The OMS Agent then sends the messages out to the Sentinel Cloud. The Syslog server in our case is RHEL 8. x running its native Rsyslog, the specifics in this document with respect to regex filtering syntax may change for other Linux/Syslog server implementations.

The default OMS agent installation created separate listening ports for Syslog formatted and CEF formatted messages, ports 25224 and 25226 respectively. Other data connectors may implement additional ports, for example, the Cisco Meraki connector implements port 22033.

Figure 1, Syslog Collection Data Flow

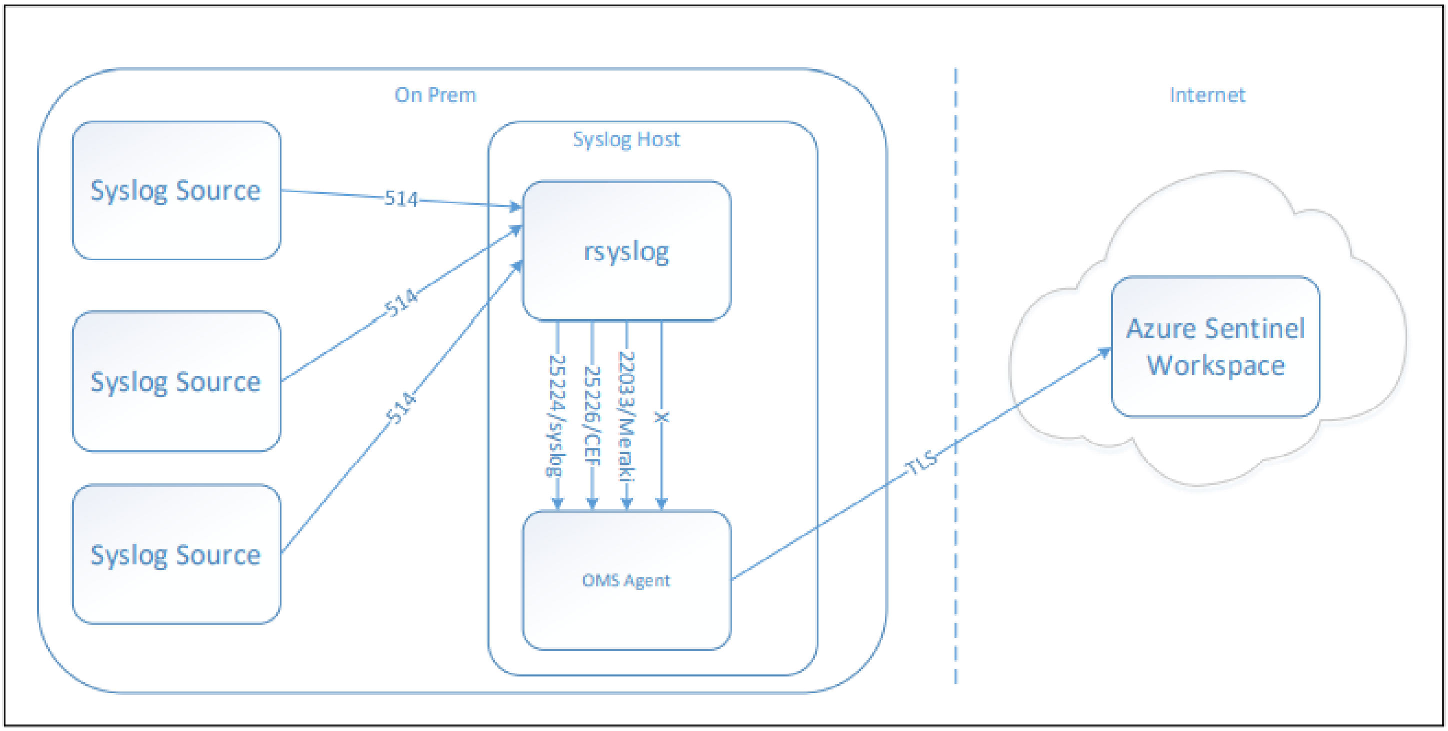

Figure 1, Syslog Collection Data Flow

- Syslog Source
- Rsyslog
- OMS Agent
- Syslog Source
- Syslog Source 25224/syslog 25226/CEF 22033/Meraki
- Azure Sentinel
- Workspace X -On-Prem Internet
- Syslog Host

III. Syslog Ingestion & Filtering Overview – RHEL 8.x/RSYSLOG

- Rsyslog filterings is implemented by simple syntax in text files. The primary file is "/etc/rsyslog.conf". The default Rsyslog implementation on RHEL 8.x put's an include statement in rsyslog.conf that points to the /ect/rsyslog.d directory where additional filtering files can be placed. Additional files are processed in alphabetical order.

- All events must be captured by a filter or they are effectively discarded.

- After a message is a capture by a filter further processing of that message must be explicitly stopped or it will be run against the next downstream filter.

- In all but the simplest of environments fairly detailed regex filtering must be used to insure that messages are consistently filtered in or out without cross-talk.

- The action for each filter-in statement will be one of the OMS agent ports, by default 25224 for traditional syslog formatted messages and 25256 for CEF.

- Some syslog type Data Connectors will add other OMS destination ports.

IV. RHEL 8.x/Rsyslog – Default Configuration Changes

The default filtering in "/etc/rsyslog.conf" in the syslog server host must be modified to prevent all of the remote messages from being capture into local files. If this is not done the local files will be quickly overrun with remote messages. This is accomplished by wrapping "if" expression syntax around whatever filter/actions are to be sent to local files only, for example:

- Change This:

```
kern.* /dev/console
*.info;mail.none;authpriv.none;cron.none /var/log/messages
authpriv.* /var/log/secure
mail.* /var/log/maillog
cron.* /var/log/cron
*.emerg :omusrmsg:*
uucp,news.crit /var/log/spooler
local7.* /var/log/boot.log
```

To This:

```
if $fromhost-ip == '127.0.0.1' then {
kern.* /dev/console
*.info;mail.none;authpriv.none;cron.none /var/log/messages
authpriv.* /var/log/secure
mail.* /var/log/maillog
cron.* /var/log/cron
*.emerg :omusrmsg:*
uucp,news.crit /var/log/spooler
local7.* /var/log/boot.log
}
```

V. Filtering Goals

- Leave no message behind.
- All of the best alerting and reporting rules are useless if messages are not being collected.
- Filter out unwanted messages.
- Prevent Cross-Talk

VI Filtering Strategy

- Whenever practical, for performance sake, configure syslog data sources to not send unwanted messages. No sense burning up network & processor bandwidth to just dump messages over the side.
- Whenever practical use TCP instead of UDP. At least in theory this would result in syslog sources queuing up messages whenever the syslog server is not available, then resuming when it's back online. Strictly

speaking UDP not a reliable transport.

- Use a single filtering file in “/etc/rsyslog.d” to effect all filtering, for example “/etc/rsyslog.d/00-filters_all.conf”.
- Disable the other filter files that are created with the default installation, namely “/etc/rsyslog.d/security-config-omsagent.conf” and “/etc/rsyslog.d/95- omsagent.conf”.
- Do not use “Home>Microsoft Sentinel>Microsoft Sentinel>Syslog>Agents Configuration” to push filtering from the Portal.
- For syslog data source types that have a clear consistent signature, employ a block of filtering in or out statements in the filtering file, examples are Cisco ASA, Cisco Meraki, and Palo Alto NGFW.
- Use expression based regex filtering, specifically by way of the “re_match” function. Simple keyword type filters cannot be specific enough to prevent cross-talk.
- The goal is to address all of a given data source type’s messages within its block.
- Syslog data source types that do not have a clear consistent signature are addressed with a “catch-all/drop-all” block at the bottom of the filter file.
- Use specific tools and methodology to develop and fine tune each filter, see “Filter Development Tools” further on.
- Order the filters such that the highest volume syslog data source type is at the top descending to the lowest.
- Place a “catch all/drop all” filter at the bottom. This serves two purposes:
- For messages that don’t have a clear enough signature it will be difficult to develop/maintain a data source type filtering block as described above. Instead these messages would just get caught by the “catch-all”.
- But before the “catch-all” these same messages should hit the “drop-all” so anything we’re certain we don’t want can be dumped.
- Monitor what messages are making it all the way down to “catch all/drop all”. This can be done by sending these messages to a local file or perhaps a different syslog server dedicated to monitoring “catch all/drop all” messages for messages that might be slipping past a filter block that should be capturing them.

VII. Rsyslog Filtering Overview

There are (3) types of filter statement structures, note that the author sadly does not understand how to effect a stop or if it is even an option for the Selector or Property based filter types. There are plenty of references out there in the world, a sampling is shown in the “References” section further on.

- Selector Based: Syntax: “Filter Action” Where the Filter has two components “Facitily.Priority” Examples:

```
kern.* /dev/console *.info;mail.none;authpriv.none;cron.none @host1.local.com
```

- Property Based: Syntax:

```
:property, [!]compare-operation, "value" Example: :msg, startswith, "val"
```

- Expression Based: Syntax:

```
if expr then action Sample: if $msg contains 'error' then /var/log/errlog & stop
```

VIII. Filter Implementation & Tuning

The process is straight forward, but even after it is “done” continual assessment must be done to assure that all messages are being properly filtered in or out.

- Capture a sample set of messages for a given data source type. This can be done with tcpdump or by opening up filtering to capture everything to a file. The challenge is to get enough of a sample to be assured that filter expressions will filter in and filter out all messages from the given syslog data source type.
- Determine what needs to be filtered in and what needs to be filtered out. This is down to figure out what messages have a security value worth the cost of ingestion.
- Develop filter-in and filter out expressions The online POSIX ERE regex testers come in handy here. The goal is to refine the regex such that it will only match the given syslog data source type (not crosstalk).
- Repeat for each syslog date source type.

IX. Filter Development Tools:

- POSIX Regex Documentation RHEL 8.x/Rsyslog supports POSIX Enhanced Regular Expressions (ERE). There are significant differences between POSIX ERE and more current familiar PCRE flavors. These are some key points:

- Meta Characters: The characters (,),[,],.,*,?,+,|,^ and $ are special symbols and have to be escaped with a backslash except we have found that the backslash does not work. Escape Character Workaround:

- At least in our RHEL 8.x/rsyslog implementation, the backslash does not work as an escape character even though documentation and the few regex POSIX ERE tester we could find imply that it is supported.

- Character Classes: Yes there are character classes but you have to put them inside an additional set of brackets to make them work. For example "ABC[:space:]123" doesn't work, it has to be "ABC[[:space:]]123"

- Bracket Expressions: Can be used to match one character from a set of characters, for example "[|]" will match "|".

- POSIX Regex Testers In fact there are not nearly as many online expression developers that we often rely on for development and testing.

This is the regex tester Rsyslog tester on the official rsyslog site:

https://www.rsyslog.com/regex/

This was the only other useful & cost free POSIX ERE tester we could find:

https://www.pagecolumn.com/tool/pregtest.htm

X. Rsyslogd config file Debugger

This command test the validity of the filtering (and other stuff), if it's not happy it will exit with something other than what's shown below.

```
rsyslogd -N 1
```

rsyslogd: version 8.1911.0-2.el8, config validation run (level 1), master config /etc/rsyslog.conf rsyslogd: End of config validation run. Bye.

XI. tcpdump

This utility is critical for understanding assessing what's inbound to the syslog server and what the syslog server is sending to the OMS ports. It's often useful to pipe the output to grep to filter it to specific text. For example, to capture any inbound traffic to port 514 that has the text "THREAT" and write that to a file: sudo tcpdump -Ani any -s0 -l port 514 | grep -i "THREAT" > tcpdump-nj-threat.txt Where the tcpdump switches are as follows: A print output in asci n Do not resolve hostnames i interface switch, in this case value is any.

XII. References:

- Rsyslog Official Documentation Site: https://www.rsyslog.com/doc
- POSIX

Meta Characters: The characters (,),[,],.,*,?,+,|,^ and $ are special symbols and have to be escaped with a backslash symbol in order to be treated as literal characters.

https://en.wikibooks.org/wiki/Regular_Expressions/POSIX-Extended_Regular_Expressions Escape Character: Like many regex implementations the escape character is a single backslash, however rsyslog seems to have a problem with this. For example "|" should match a literal "|'. We have found that filters the use this syntax work but rsyslog then fails to load subsequent filters. A workaround is to use a character class "[|]".

Bracket Expressions: https://www.regularexpressions.info/posixbrackets.html#:~:text=POSIX bracket expressions are a,start negates the bracket expression.

- Expression Testers: This is the regex tester Rsyslog tester on the official rsyslog site:

https://www.rsyslog.com/regex/

This was the only other useful & cost free POSIX ERE tester we could find:

https://www.pagecolumn.com/tool/pregtest.htm

- Syslog Latest RFC 5424:

https://datatracker.ietf.org/doc/html/rfc5424#:~:text=RFC 5424 - The Syslog Protocol

- Basic Message Structure: Where RFC defines a detailed structure it's more common to find syslog messages the only have the first few parts of the header and then it's a free for all. For example in the Cisco ASA message below we see the priority, timestamp, but no hostname. When the hostname is missing many syslog servers will assume the messages source IP to be the host.

```
<190>Apr 11 2022 16:32:57: %ASA-6-302013: Built inbound TCP connection 2968983820 for
DMZ:10.1.40.1/66408 (10.1.40.1/55408) to XXXXXXXXXX:10.18.190.50/389 (10.1.90.20/342)
```

- CEF: The Common Event Format was perhaps a better solution to the syslog free for all format, but has not been widely adopted or enshrined into an RFC. CEF messages have a header followed by any number of name value pairs in the "Extension" field.

```
CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension
```

The name value pairs tend to be built around a prefix of device, source and destination. The device prefix tag the message originator, source and destination are then used to tag the “actor” and thing “acted on” or “traget”. For example admin X (actor) changed user Y (target).

https://kc.mcafee.com/resources/sites/MCAFEE/content/live/CORP_KNOWLEDGEBASE/78000/KB78712/en_US/CEF_White_Paper_20100722.pdf

- Facility/Severity to Priority: The facility and severity are combined into a priority shown in the figure below. The calculation is as follows: (Facility Value * 8) + Severity Value = PRI. Figure 2, Syslog Facility/Severity to Priority

XIII. Sample Filter File

A sample filter file is shown below, there are some key things to highlight here: All of the “if” statements should be on a single line, the page width limits are causing to wrap. All such lines have been highlighted. The “& stop” must be on a separate line. The “re_match()” function required that the string be in double quotes, for example:

```
If re_match(“cat”) then stop
#Cisco ASA
#Messages are sent in sysmess but are converted by the OMS agent
##Filter in
if re_match($rawmsg,"[[:digit:]]{2}:[[:digit:]]{2}:[[:digit:]]{2}:[[:space:]]%ASA-[[:digit:]]{1,4}-
[[:digit:]]{4,10}") then @@127.0.0.1:25226
& stop
#Palo Alto NGFW,IDS,IPS Filter
#Messages are sent in native CEF
##Filter in
if re_match($rawmsg,"CEF:[[:digit:]]*.Palo[[:space:]]Alto[[:space:]]Networks.PAN-OS.") then
@@127.0.0.1:25226
& stop
#Cisco Meraki
##Filter in
if re_match($rawmsg,"[[:digit:]]{10}.[[:digit:]]{6,9}[[:space:]].*_RTR.*[[:space:]](urls|ids alerts|events|l7_
& stop
##Filter Out
if
re_match($rawmsg,"[[:digit:]]{10}.[[:digit:]]{6,9}([[:space:]].*[[:space:]]|.?)(flows|ip_flow_start|ip_flow_
end)") then stop
#Catch All
##Filter out
```

- Because the catch-all filter below this will grab everything, this is the last chance to fitler things out. Filter In The purpose of this filter is to catch anything that slips by all other filters.

```
if re_match($rawmsg,".") then @127.0.0.1:25224
& stop
```