corelight

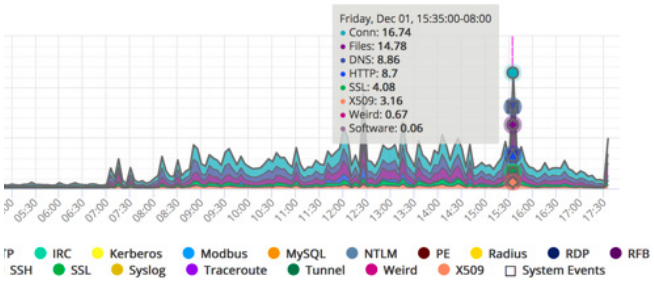
Tuning your log volume.

The Corelight Sensor produces an astonishing amount of useful network telemetry. However, it can become a victim of its own success, unintentionally filling up disks or driving up SIEM costs. You can tune your sensor in various ways to reduce sensitivity, while avoiding accidentally blinding yourself to an active adversary in your network. This paper will review in detail some techniques you can use to tune your log volume.

First: measure

Once you have traffic hitting your monitoring ports, you can check out the expected log volumes—you don't even need to be exporting them anywhere. This is easiest to see in the web interface. To check it out, first enable the web interface under Access in the UI, then go to the sensor in your web browser, log in, and click on the stats link (the link says "click here"). The top graph on the new page will show you expected log volume (see figure below).

As you hover over the graph, a popup will give you the rates for the sampling period for each log type. The shape of the graph for a 24 hour period will also help you estimate how many hours are active vs. idle. You can enable/disable specific log types by clicking on them in the footer of the graph.



The different colors represent different logs. The y-axis shows log volume rate over 5 minute intervals and the popup (grey) the actual values for logs during that interval.

Split logging

One fundamental technique our customers use is split logging. In split logging, you send some of your logs streaming in real time to your analysis platform (e.g., Splunk, Elastic Search) but store the bulk of them as files on inexpensive storage. You then do the majority of your incident response on the SIEM, but have the full logs to go back to as needed on the server. Over time, you rotate the logs off to backup. Some organizations have years of logs stored this way to check retroactively for incident analysis.

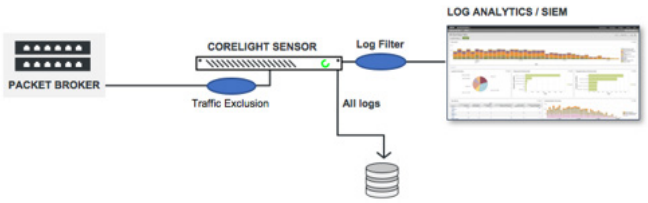
The advantage of using split logging is you do not lose information fidelity. If an incident turns up the need for log info that is on the server, you either search for it there, or ingest those logs into the SIEM when you need them. The logs you select for real-time incident analysis have the data you typically use.

Disabling specific logs

Either as part of creating split logging, or on its own, you may want to disable specific logs. Beyond splitting, other reasons typically include logs you simply do not find of interest (weird.log for example) or that you already collect another way (e.g., DHCP.log or syslog.log). You can control the logs you omit by listing them in the UI in the Export section under "Zeek logs to exclude" for the exporter where you want to omit them (Splunk, Kafka, syslog, JSON over TCP, SFTP).You can also change these on the fly with the corelight-client, e.g.:

corelight-client configuration update --zeek.export.splunk.exclude "weird,syslog,dhcp"

Note you are setting the whole list each time and not adding/removing specific ones. You can use this facility to automate actions or as part of the investigation recipes in your orchestration system.



Network traffic can be dropped with a Traffic Exclusion rule at input. Logs can be split and all go to storage (over SFTP or S3) with a reduced stream sent in real-time into a SIEM.

Wholesale traffic exclusion

Due to the placement of the Corelight Sensor within the network, it may receive traffic that you do not want to analyze at all. You can use the input filter, set via Localize->Traffic exclusion filter or **zeek.site.ignore_bpf** in the client to remove that traffic.

Supposing for example that the whole 192.168 subnet is uninteresting for your analysis, you can use the filter "net 192.168.0.0/16" to drop it from all logs. Or drop all UDP or all GRE, and so on. Since this will completely remove visibility of the traffic, construct your filters with care! You may want to refer to the Manual (section *Traffic Exclusion Filters*) and the Knowledge Base (article *Additional Filter Examples*) for syntax and examples.

Another example of traffic you might want to ignore is broadcast or multicast traffic, as it is not a good mechanism for attackers, e.g.:

net 224.0.0.0/4 OR net FF00::/8

Additionally, where you have internal hosts whose task it is to scan your network, you probably don't want to record the logs for their traffic. You may also have specific hosts like online backup systems or patch servers with frequent connections internally. You can remove them by ip:port combinations. If your vantage point shows both client->DNS server and DNS server->external resolver traffic, you may want to deduplicate by removing the latter.

Enable data reduction package

Corelight's Data Reduction Package is included in the collection of pre-installed packages and reduces the data volume of common log types by suppressing typically low-value and duplicate log entries. This can be very easily enabled, and frequently reduces the volume of data by about 30% with minimal impact on network visibility. It is important to review the reduced log descriptions below, however, so you understand what event records are being removed.

For each supported log type the package introduces an alternative version, named after the original log and suffixed with "_red", that contains the reduced log stream. The idea here is that for some amount of time you may want to consume both the original and the reduced log streams, to build up familiarity. You can suppress the original log streams as usual, via the log exclusion configuration on your exporters.

The package also introduces datared.log, a statistics log that compares the number of log entries in the unreduced to that of the reduced streams, regardless of whether the original logs currently get exported.

Depending on the log type at hand, the reduced version has exactly the same column types as the unreduced version, or minor variations. However, there is some loss in information contained in the logs. When you first enable Data reduction there is a slight delay to apply the change of approximately 90 seconds. After that, you should immediately start receiving *conn_red*, *http_red* and *ssl_red*. It will be about 10 minutes before you start seeing *dns_red* and *files_red.*

In the following we walk through the supported logs, each of which you can enable/disable in the package's configuration section, and describe the reductions applied. All reductions are enabled by default when you enable the package.

conn.log

When enabled, the package omits "unproductive" flows in which payload travels only in one direction, and omits DNS and NetBIOS flows (queries to UDP ports 53 or 137, respectively). conn_red.log looks identical to conn.log.

dns.log

When enabled, the package omits duplicate entries in a 10-millisecond sliding window (a timeframe sufficient to catch duplicates in the traffic we studied for this feature), omits NetBIOS wildcard queries ("*"), and broken entries containing neither queries nor answers (usually a result of tapping problems).

dns_red

Field	Description
ts	The earliest time at which a DNS protocol message over the associated connection is observed.
uid	A unique identifier of the connection over which DNS messages are being transferred.
id	The connection's 4-tuple of endpoint addresses/ports.
query	The domain name that is the subject of the DNS query.
qtype_ name	A descriptive name for the type of the query.
answers	The set of resource descriptions in the query answer.
num	How often we've seen this query in this coalescence interval.

files.log

When enabled, the package omits duplicate files in a 10-minute sliding window. Note that the actual *extraction* of files, if configured, continues to operate independently of this log-reduction feature.

files_red

Field	Description
ts	The time(s) when the file was seen.
fuid	An identifier associated with one of the highly similar entries. We use the first fuid seen within the window for this.
tx_hosts	The originating hosts involved in the transfers.
rx_hosts	The responding hosts involved in the transfers.
conn_uids	Connection UIDs over which the file was transferred.
source	An identification of the source of the file data. E.g., it may be the network protocol over which it was transferred, or a local file path which was read, or some other input source.
depth	A value to represent the depth of this file in relation to its source. In SMTP, it is the depth of the MIME attachment on the message. In HTTP, it is the depth of the request within the TCP connection.
analyzers	A set of analysis types done during the file analysis.
mime_type	A mime type provided by the strongest file magic signature match against the *bof_buffer* field of fa_file, or in the cases where no buffering of the beginning of file occurs, an initial guess of the mime type based on the first data seen.
filename	A filename for the file if one is available from the source for the file. These will frequently come from "Content-Disposition" headers in network protocols.

Field	Description
local_orig	If the source of this file is a network connection, this field indicates if the data originated from the local network or not as determined by the configured Site::local_nets.
is_orig	If the source of this file is a network connection, this field indicates if the file is being sent by the originator of the connection or the responder.
seen_bytes	Number of bytes provided to the file analysis engine for the file.
total_bytes	Total number of bytes that are supposed to comprise the full file.
missing_bytes	The number of bytes in the file stream that were completely missed during the process of analysis e.g. due to dropped packets.
overflow_bytes	The number of bytes in the file stream that were not delivered to stream file analyzers. This could be overlapping bytes or bytes that couldn't be reassembled.
timedout	Whether the file analysis timed out at least once for the file.
parent_fuid	Identifier associated with a container file from which this one was extracted as part of the file analysis.
extracted	Local filename of extracted file.
extracted_cutoff	Set to true if the file being extracted was cut off so the whole file was not logged.
extracted_size	The number of bytes extracted to disk.
md5	An MD5 digest of the file contents.
sha1	A SHA1 digest of the file contents.
sha256	A SHA256 digest of the file contents.
num	Number of times we've seen this file.

http.log

When enabled, the package omits proxied connections. http_red.log looks identical to http.log.

ssl.log

When enabled, the package omits duplicate certificates in a 10-minute sliding window, and omits valid certificates. ssl_red.log looks identical to ssl.log.

weird.log

When enabled, the package omits the following common but low-value weird types:

- dns_unmatched_msg
- dns_unmatched_msg_quantity
- dns_unmatched_reply

datared

Field	Description
ts	The time at which Zeek reported this set of statistics.
conn_red	The reduced number of conn.log entries.
conn_total	The original number of conn.log entries.
dns_red	The reduced number of DNS.log entries.
dns_total	The original number of DNS.log entries.
dns_coal_miss	The number of times we wanted to store additional DNS log entries for coalescence, but exceeded the storage budget.
files_red	The reduced number of files.log entries.
files_total	The original number of files.log entries.
extracted	Local filename of extracted file.
files_coal_miss	The number of times we wanted to store additional files log entries for coalescence, but exceeded the storage budget.
http_red	The reduced number of HTTP.log entries.
http_total	The original number of HTTP.log entries.
ssl_red	The reduced number of SSL.log entries.
ssl_total	The original number of SSL.log entries.
ssl_coal_miss	The number of times we wanted to store additional SSL log entries for coalescence, but exceeded the storage budget.
weird_red	The reduced number of weird.log entries.
weird_total	The original number of weird.log entries.

datared.log

This log provides the number of log entries that would end up getting reported to the original log, compared to the number of entries in the reduced logs. This allows you to compute data-reduction ratios at line granularity for your network.

Custom log line exclusion

You can leverage the analyzed data for your filters with output filters. These are set in Export->Export Zeek Streaming Logs->Log filter or in the client zeek.export. logs.filter. Note that at present they only apply to the streaming logs. Having the fully analyzed information grants powerful ability to choose what to drop. Some of these implement ideas also present in the log reduction package.

Scans

Users, security infrastructure, and attackers can create a lot of logs while performing network scans. The conn log provides a log field useful for dropping them, the connection state (conn_state). For predominantly one-sided traffic, like scans, you expect either an unanswered SYN packet, or a rejected one, so you can look for states of S0, RST0, REJ or RST0S0 in the conn log:

> ***$log = "conn" and ((conn_state = "S0" or conn_state = "RSTO" or conn_state = "RSTOS0" or conn_state = "REJ"))***

This will remove all scans (including internal security devices, some software misconfigurations and other recon activity). Maybe you prefer to locate local hosts who are scanning, and using a more concise notion, you could refine your filter to:

> ***$log = "conn" and (history in ["S", "Sr"] and local_orig=false)***

You could further refine to exclude local security devices that scan (though that might be more appropriate in an input filter).

Reducing DNS logging

The DNS log is powerful, providing a way to link DNS requests with responses at enterprise scale. However, in large environments it will see a lot of traffic. One trivial change is to not track local resolution, so the site example.com might add a filter:

> ***$log = "dns" and (qtype_name = "A" AND ".example.com" in query)***

Some devices also can frequently issue blank DNS requests, so if you have them, you might extend the filter to:

> ***$log = "dns" AND qtype_name="A" AND (".example.com" in query OR query = "")***

We can refine the filter to handle IPv6 and tighten it up to require .com to be at the end of the string (so, for example, we don't exclude a name like www.example.com.hackershome.com from being logged):

> ***$log = "dns" AND (qtype_name in ["A","AAAA"] AND (query ~ /\.example\.com$/i OR query = ""))***

Another option is to keep logging local lookups that fail with NXDOMAIN, so you can catch reconnaissance:

> ***$log = "dns" AND (qtype_name in ["A","AAAA"] AND (query ~ /\.example\.com$/i OR query = "") AND rcode!=3)***

Pointer DNS records

Pointer records are for reverse lookups. Locally answered requests for RFC 1918 addresses (or for locally answered public address space, for those organizations with it) are common in the logs, but might not have much utility for security analysis:

> ***$log = "dns" AND qtype=12 AND query ~ /(\.10|168\.192|((1[6789]|2\d|30|31)\.172))\.in-addr\.arpa$/i***

If you were sufficiently concerned, you could further restrict the filter to only exclude answers provided by your own DNS servers; otherwise a compromised machine could set its DNS to an external server (for exfiltration via covert communication) and you would be blind to its PTR lookups in this space.

Often DNS logs have a lot of .local (multicast DNS) and WORKGROUP queries you might wish to remove:

> ***$log="DNS" AND query="WORKGROUP"***
>
> ***$log="DNS" AND query ~ /\.local$/***

In the latter example above, you might instead have removed this traffic by excluding multicast DNS with an input filter, but if handling here, you might exclude these from the conn log as well:

$log="conn" AND id.orig_p=5353 AND id.resp_p=5353 AND id.resp_h in [224.0.0.251,ff02::fb]

Conn logs for DNS traffic

Since the DNS log has all the information you need about connections that are DNS requests, their conn log entries don't add any particular value, so that's an opportunity to tune them out:

$log="conn" AND service="dns"

You wouldn't want to filter just on UDP+port as then attackers could tunnel non-DNS that way, but since the connection is detected as DNS, you can rely on the DNS log.

Certificate files

The Corelight Sensor treats certificates as "files" in addition to their SSL log record. This enables the powerful file extraction capabilities on certificates. However, most cert information is in the ssl log, so the files log entry is not useful. You remove it with:

$log = "files" AND mime_type="application/pkix-cert"

High volume sites

There may be some high traffic volume sites that you're willing to trust. Exactly which will depend on your environment:

$log="dns" AND (query ~ /monitoring\.amazonaws\.com$/i or

query ~ /\.google(apis)?\.com/i or

query ~ (facebook|hipchat|meraki|mozilla|apple|gmail)\.com$/i or

query ~ /(ntp|fedoraproject)\.org$/i)

You can generate your own list by doing a frequency analysis on your own DNS logs.

Changing your Corelight Sensor configuration

When using multiple filters, be careful to combine them properly. The filter option on the Corelight Sensor identifies what to exclude. If you have multiple filters, you should combine them with OR, because you want traffic that matches any of them to be excluded. You should show grouping by putting parenthesis around each, e.g.,:

($log="dns" and query="WORKGROUP") or ($log = "files" AND mime_type="application/pkix-cert")

While strictly speaking the parentheses are not needed for proper parsing, they can aid in human understanding of intent or when editing in the UI. If you are putting the filters into the UI directly, we recommend recording the individual filters outside the Corelight Sensor in a document, so you can make sense of the filter the next time you change it. A better solution is to set the filter via a script that uses the corelight-client—that way you can use features of the shell to document the why's of the rules:

```
#!/bin/bash
# skip workgroup messages in DNS log
export dns='$log="dns" AND
query="WORKGROUP"'
# do not record files for certs
export files='$log = "files" AND mime_
type="application/pkix-cert"'
corelight-client configuration update --zeek.
export.logs.filter "($dns) or ($files)"
```

Checking with Corelight

Here at Corelight we're focused on your success as a customer, and are happy to help you craft these filters if you would like to bounce ideas off of us. Your Sales Engineer, Technical Account Manager, or the Support TAC can help. Your tips may contribute to a later edition of this paper, or feature in the Knowledge Base. Several of the current examples came from discussions with customers.

Conclusion

Filtering can provide a big win for reducing your log volume. Split logging is a solid backstop in case you accidentally remove more logs than desired or in cases where an adversary finds a blind spot in your filters. Log volume can be safely reduced by 50% by using the techniques described here.

corelight

corelight

Corelight delivers the most powerful network security monitoring (NSM) solutions that help large organizations defend themselves by transforming network traffic into rich logs, extracted files, and security insights. Corelight makes a family of virtual, cloud and physical sensors that take the pain out of deploying open-source Zeek and make it faster and enterprise-ready. Corelight's customers include Fortune 500 companies, government agencies, and research universities.

We make the **world's networks safer.**

CORELIGHT, INC. | INFO@CORELIGHT.COM | CSD001-TLV-V2.0-US

Contact us

For more information or to schedule an evaluation:

info@corelight.com

888-547-9497

510-281-0760

corelight.com