corelight

White Paper

Introduction to Zeek log formats and review

Zeek creates a variety of logs when run in its default configuration. This data can be intimidating for a first-time user. In this section, we will process a sample packet trace with Zeek, and take a brief look at the sorts of logs Zeek creates. We will look at logs created in the traditional format, as well as logs in a newer format. We will also introduce a few command-line tools to examine Zeek logs. We worked with this sample data on a Linux system.

Working with a Sample Trace

For the examples that follow, we will use Zeek on a Linux system to process network traffic captured and stored to disk. We saved this trace file earlier in packet capture (PCAP) format as tm1t.pcap. The command line protocol analyzer Tcpdump, which ships with most Unix-like distributions, summarizes the contents of this file.

```
zeek@zeek:~/zeek-test$ tcpdump -n -r tm1t.pcap

reading from file tm1t.pcap, link-type EN10MB (Ethernet)
14:39:59.305988 IP 192.168.4.76.36844 > 192.168.4.1.53: 19671+ A?
testmyids.com. (31)
14:39:59.306059 IP 192.168.4.76.36844 > 192.168.4.1.53: 8555+ AAAA?
testmyids.com. (31)
14:39:59.354577 IP 192.168.4.1.53 > 192.168.4.76.36844: 8555 0/1/0 (94)
14:39:59.372840 IP 192.168.4.1.53 > 192.168.4.76.36844: 19671 1/0/0 A
31.3.245.133 (47)
```

```
14:39:59.430166 IP 192.168.4.76.46378 > 31.3.245.133.80: Flags [S], seq
3723031366, win 65535, options [mss 1460,sackOK,TS val 3137978796 ecr
0,nop,wscale 11], length 0
14:39:59.512232 IP 31.3.245.133.80 > 192.168.4.76.46378: Flags [S.], seq
2993782376, ack 3723031367, win 28960, options [mss 1460,sackOK,TS val
346747623 ecr 3137978796,nop,wscale 7], length 0
14:39:59.512284 IP 192.168.4.76.46378 > 31.3.245.133.80: Flags [.], ack 1, win
32, options [nop,nop,TS val 3137978878 ecr 346747623], length 0
14:39:59.512593 IP 192.168.4.76.46378 > 31.3.245.133.80: Flags [P.], seq 1:78,
ack 1, win 32, options [nop,nop,TS val 3137978878 ecr 346747623], length 77:
HTTP: GET / HTTP/1.1
14:39:59.600488 IP 31.3.245.133.80 > 192.168.4.76.46378: Flags [.], ack 78, win
227, options [nop,nop,TS val 346747711 ecr 3137978878], length
```

This is a simple exchange involving domain name system (DNS) traffic followed by HyperText Transfer Protocol (HTTP) traffic.

Rather than run Zeek against a live interface, we will ask Zeek to digest this trace. This process allows us to vary Zeek's run-time operation, keeping the traffic constant.

First we make two directories to store the log files that Zeek will produce. Then we will move into the "default" directory.

```
zeek@zeek:~/zeek-test$ mkdir default
zeek@zeek:~/zeek-test$ mkdir json
zeek@zeek:~/zeek-test$ cd default/
```

Zeek TSV Format Logs

From this location on disk, we tell Zeek to digest the tm1t.pcap file. The -C flag tells Zeek to ignore any TCP checksum errors. This happens on many systems due to a feature called "checksum offloading," but it does not affect our analysis. The -r flag tells Zeek where to find the trace of interest.

```
zeek@zeek:~/zeek-test/default$ zeek -C -r ../tm1t.pcap
```

Zeek completes its task without reporting anything to the command line. This is standard Unix-like behavior. Using the ls command we see what files Zeek created when processing the trace.

```
zeek@zeek:~/zeek-test/default$ ls -al

total 28
drwxrwxr-x 2 zeek zeek 4096 Jun 5 14:48 .
drwxrwxr-x 4 zeek zeek 4096 Jun 5 14:43 ..
-rw-rw-r-- 1 zeek zeek  737 Jun 5 14:48 conn.log
-rw-rw-r-- 1 zeek zeek  778 Jun 5 14:48 dns.log
-rw-rw-r-- 1 zeek zeek  712 Jun 5 14:48 files.log
-rw-rw-r-- 1 zeek zeek  883 Jun 5 14:48 http.log
-rw-rw-r-- 1 zeek zeek  254 Jun 5 14:48 packet_filter.log
```

Zeek created five files. We will look at the contents of Zeek log data in detail in later sections. For now, we will take a quick look at each file, beginning with the packet_filter.log. This log shows any filters that Zeek applied when processing the trace. We use the cat command to show the contents of each log.

```
zeek@zeek:~/zeek-test/default$ cat packet_filter.log

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path packet_filter
#open 2020-06-05-14-48-32
#fields ts node filter init success
#types time string string bool bool
1591368512.420771 zeek ip or not ip T T
```

Next we look at Zeek's conn.log.

```
zeek@zeek:~/zeek-test/default$ cat conn.log

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path conn
```

```
#open 2020-06-05-14-48-32
#fields ts uid id.orig_h id.orig_p id.resp_h
id.resp_p proto service duration orig_bytes resp_bytes
conn_state local_orig local_resp missed_bytes history orig_pkts
orig_ip_bytes resp_pkts resp_ip_bytes tunnel_parents
#types time string addr port addr port enum string
interval count count string bool bool count string count
countcount count set[string]
1591367999.305988 CazOhH2qDUiJTWMCY 192.168.4.76 36844
192.168.4.1 53 udp dns 0.066852 62 141 SF
- -0 Dd 2 118 2 197 -
1591367999.430166 CLqEx41jYPOdfHF586 192.168.4.76 46378
31.3.245.133 80 tcp http 0.254115 77 295 SF
- -0 ShADadFf 6 397 4 511 -
#close 2020-06-05-14-48-32
```

Next we look at Zeek's dns.log.

```
zeek@zeek:~/zeek-test/default$ cat dns.log

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path dns
#open 2020-06-05-14-48-32
#fields ts uid id.orig_h id.orig_p id.resp_h
id.resp_p proto trans_id rtt query qclass qclass_name
qtypeqtype_name rcode rcode_name AA TC RD RA Z
answers TTLs rejected
#types time string addr port addr port enum count
interval string count string count string count string bool
bool bool bool count vector[string] vector[interval] bool
1591367999.306059 CazOhH2qDUiJTWMCY 192.168.4.76 36844
192.168.4.1 53 udp 8555 - testmyids.com 1
C_INTERNET 28 AAAA 0 NOERROR F F T F 0
- - F
1591367999.305988 CazOhH2qDUiJTWMCY 192.168.4.76 36844
192.168.4.1 53 udp 19671 0.066852 testmyids.com 1
C_INTERNET 1 A 0 NOERROR F F T T 0
```

```
31.3.245.133 3600.000000 F
#close 2020-06-05-14-48-32
```

Next we look at Zeek's files.log.

```
zeek@zeek:~/zeek-test/default$ cat files.log

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path files
#open 2020-06-05-14-48-32
#fields ts fuid tx_hosts rx_hosts conn_uids source
depth analyzers mime_type filename duration
local_orig is_orig seen_bytes total_bytes missing_bytes
overflow_bytes timedout parent_fuid md5 sha1 sha256
extracted extracted_cutoff extracted_size
#types time string set[addr] set[addr] set[string] string
count set[string] string string interval bool bool
countcount count count bool string string string string string
bool count
1591367999.604000 FEEsZS1w0Z0VJIb5x4 31.3.245.133 192.168.4.76
CLqEx41jYPOdfHF586 HTTP 0 (empty) text/plain -
0.000000 - F 39 39 0 0 F - -
- - - - -
#close 2020-06-05-14-48-32
```

Finally we look at Zeek's http.log.

```
zeek@zeek:~/zeek-test/default$ cat http.log

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path http
```

```
#open 2020-06-05-14-48-32
#fields ts uid id.orig_h id.orig_p id.resp_h
id.resp_p trans_depth method host uri referrer version
user_agent origin request_body_len response_body_len
status_code status_msg info_code info_msg tags
username password proxied orig_fuids orig_filenames
orig_mime_types resp_fuids resp_filenames resp_mime_types
#types time string addr port addr port count string string
string string string string string count count count string
countstring set[enum] string string set[string] vector[string]
vector[string] vector[string] vector[string] vector[string] vector[string]
1591367999.512593 CLqEx41jYPOdfHF586 192.168.4.76 46378
31.3.245.133 80 1 GET testmyids.com / - 1.1
curl/7.47.0 - 0 39 200 OK - - (empty) -
- - - - - FEEsZS1w0Z0VJIb5x4 -
text/plain
#close 2020-06-05-14-48-32
```

As we can see with each log file, there is a set of headers beginning with the hash character (#) followed by metadata about the trace. This format is the standard version of Zeek data, represented as tab separated values (TSV).

Interpreting this data as shown requires remembering which “column” applies to which “value.” For example, in the dns.log, the third field is id.orig_h, so when we see data in that field, such as 192.168.4.76, we know that 192.168.4.76 is id.orig_h.

One of the common use cases for interacting with Zeek log files requires analyzing specific fields. Investigators may not need to see all of the fields produced by Zeek when solving a certain problem. The following sections offer a few ways to address this concern when processing Zeek logs in text format.

Zeek TSV Format and Awk

A very traditional way of interacting with Zeek logs involves using native Unix-like text processing tools like awk. Awk requires specifying the fields of interest as positions in the log file. Take a second look at the dns.log entry above, and consider the parameters necessary to view only the source IP address, the query, and the response. These values appear in the 3rd, 10th, and 22nd fields in the Zeek TSV log entries. Therefore, we could invoke awk using the following syntax:

```
zeek@zeek:~/zeek-test/default$ awk '/^[^#]/ {print $3, $10, $22}' dns.log

192.168.4.76 testmyids.com -
192.168.4.76 testmyids.com 31.3.245.133
```

Now we have a much more compact log, with just the fields we want. Unfortunately, this requires specifying fields by location. If we were to modify the log output, or if the Zeek project were to change the log output, any scripts we built using awk and field locations would require modification. For this reason, the Zeek project recommends alternatives like the following.

Zeek TSV Format and Zeek-Cut

The Zeek project provides a tool called zeek-cut to make it easier for analysts to interact with Zeek logs in TSV format. Zeek-cut parses the header in each file and allows the user to refer to the specific columnar data available. This is in contrast to tools like awk that require the user to refer to fields referenced by their position.

Consider the dns.log generated earlier. If we process it with zeek-cut, without any modifications, this is the result:

```
zeek@zeek:~/zeek-test/default$ cat dns.log | zeek-cut

1591367999.306059 CazOhH2qDUiJTWMCY 192.168.4.76 36844
192.168.4.1 53 udp 8555 - testmyids.com 1
C_INTERNET 28 AAAA 0 NOERROR F F T F 0
- - F
1591367999.305988 CazOhH2qDUiJTWMCY 192.168.4.76 36844
192.168.4.1 53 udp 19671 0.066852 testmyids.com 1
C_INTERNET 1 A 0 NOERROR F F T T 0
31.3.245.133 3600.000000 F
```

That is the dns.log, minus the header fields showed earlier. Note we have to invoke the cat utility in a pipeline to process files with zeek-cut.

If we pass zeek-cut the fields we wish to see, the output looks like this:

```
zeek@zeek:~/zeek-test/default$ cat dns.log | zeek-cut id.orig_h query answers

192.168.4.76 testmyids.com -
192.168.4.76 testmyids.com 31.3.245.133
```

The sequence of field names given to zeek-cut determines the output order. This means you can also use zeek-cut to reorder fields. For example:

```
zeek@zeek:~/zeek-test/default$ cat dns.log | zeek-cut query answers id.orig_h

testmyids.com - 192.168.4.76
testmyids.com 31.3.245.133 192.168.4.76
```

This feature can be helpful when piping output into programs like “sort.”

Zeek-cut uses output redirection through the cat command and | operator. Whereas tools like awk allow you to indicate the log file as a command line option, zeek-cut only takes input through redirection such as | and <.

For example, instead of using cat and the pipe redirector, we could obtain the previous output with this syntax:

```
zeek@zeek:~/zeek-test/default$ zeek-cut id.orig_h query answers < dns.log

192.168.4.76 testmyids.com -
192.168.4.76 testmyids.com 31.3.245.133
```

Note that in its default setup, watching a live interface and writing logs to disk, Zeek will rotate log files on an hourly basis. Zeek will move the current log file into a directory named using the format YYYY-MM-DD. Zeek will use gzip to compress the file with a naming convention that includes the log file type and time range of the file.

When processing a compressed log file, use the zcat tool instead of cat to read the file. Consider working with the gzip-encoding file created in the following example. For demonstration purposes, we create a copy of the dns.log file as dns1.log, gzip it, and then read it with zcat instead of cat.

```
so16@so16:~/zeek-test/default$ cp dns.log dns1.log

so16@so16:~/zeek-test/default$ gzip dns1.log
so16@so16:~/zeek-test/default$ zcat dns1.log.gz

#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path dns
#open 2020-06-05-14-48-32
#fields ts uid id.orig_h id.orig_p id.resp_h
id.resp_p proto trans_id rtt query qclass qclass_name
qtypeqtype_name rcode rcode_name AA TC RD RA Z
answers TTLs rejected
#types time string addr port addr port enum count
interval string count string count string count string bool
bool bool bool count vector[string] vector[interval] bool
1591367999.306059 CazOhH2qDUiJTWMCY 192.168.4.76 36844
192.168.4.1 53 udp 8555 - testmyids.com 1
C_INTERNET 28 AAAA 0 NOERROR F F T F 0
- - F
1591367999.305988 CazOhH2qDUiJTWMCY 192.168.4.76 36844
192.168.4.1 53 udp 19671 0.066852 testmyids.com 1
C_INTERNET 1 A 0 NOERROR F F T T 0
31.3.245.133 3600.000000 F
#close 2020-06-05-14-48-32
```

Zeek-cut accepts the flag -d to convert the epoch time values in the log files to human-readable format. For example, observe the default timestamp value:

```
zeek@zeek:~/zeek-test/default$ cat dns.log | zeek-cut ts id.orig_h query
answers

1591367999.306059 192.168.4.76 testmyids.com -
1591367999.305988 192.168.4.76 testmyids.com 31.3.245.133
```

Now see the effect of using the -d flag:

```
zeek@zeek:~/zeek-test/default$ cat dns.log | zeek-cut -d ts id.orig_h query answers

2020-06-05T14:39:59+0000 192.168.4.76 testmyids.com -
2020-06-05T14:39:59+0000 192.168.4.76 testmyids.com 31.3.245.133
```

Converting the timestamp from a log file to UTC can be accomplished with the -u option.

The default time format when using the -d or -u is the strftime format string %Y-%m-%dT%H:%M:%S%z which results in a string with year, month, day of month, followed by hour, minutes, seconds and the timezone offset.

The default format can be altered by using the -D and -U flags, using the standard strftime syntax. For example, to format the timestamp in the US-typical "Middle Endian" you could use a format string of: %d-%m-%YT%H:%M:%S%z

```
zeek@zeek:~/zeek-test/default$ cat dns.log | zeek-cut -D %d-%m-%YT%H:%M:%S%z
ts id.orig_h query answers

05-06-2020T14:39:59+0000 192.168.4.76 testmyids.com -
05-06-2020T14:39:59+0000 192.168.4.76 testmyids.com 31.3.245.133
```

Using awk and zeek-cut have been the traditional method of interacting with Zeek logs. In the next section we will look at the possibilities once we enable an alternative output format.

Zeek JSON Format Logs

During the last decade, the Javascript Object Notation (JSON) format has become a standard way to label and store many types of data. Zeek offers support for this format. In the following example we will re-run the tm1t.pcap trace through Zeek, but request that it output logs in JSON format.

First we change into the json directory to avoid overwriting our existing log files.

```
zeek@zeek:~/zeek-test/default$ cd ../json/
```

Next we tell Zeek to output logs in JSON format using the -e switch and the command as shown.

```
zeek@zeek:~/zeek-test/json$ zeek -C -e 'redef LogAscii::use_json=T;' -r ../tm1t.pcap
```

When we look at the directory contents, we see the same five output files.

```
zeek@zeek:~/zeek-test/json$ ls -al
```

```
total 28
drwxrwxr-x 2 zeek zeek 4096 Jun 5 14:47 .
drwxrwxr-x 4 zeek zeek 4096 Jun 5 14:43 ..
-rw-rw-r-- 1 zeek zeek  708 Jun 5 14:47 conn.log
-rw-rw-r-- 1 zeek zeek  785 Jun 5 14:47 dns.log
-rw-rw-r-- 1 zeek zeek  325 Jun 5 14:47 files.log
-rw-rw-r-- 1 zeek zeek  405 Jun 5 14:47 http.log
-rw-rw-r-- 1 zeek zeek   90 Jun 5 14:47 packet_filter.log
```

However, if we look at the file contents, the format is much different.

First we look at the packet filter log.

```
zeek@zeek:~/zeek-test/json$ cat packet_filter.log

{"ts":1591368442.854585,"node":"zeek","filter":"ip or not
ip","init":true,"success":true}
zeek@zeek:~/zeek-test/json$ cat conn.log
{"ts":1591367999.305988,"uid":"CMdzit1AMNsmfAIiQc","id.orig_h":"192.168.4.76","
id.orig_p":36844,"id.resp_h":"192.168.4.1","id.resp_p":53,"proto":"udp","servic
e":"dns","duration":0.06685185432434082,"orig_bytes":62,"resp_bytes":141,"conn_
state":"SF","missed_bytes":0,"history":"Dd","orig_pkts":2,"orig_ip_bytes":118,"
resp_pkts":2,"resp_ip_bytes":197}
{"ts":1591367999.430166,"uid":"C5bLoe2Mvxqhawzqqd","id.orig_h":"192.168.4.76","
id.orig_p":46378,"id.resp_h":"31.3.245.133","id.resp_p":80,"proto":"tcp","service":"http",
"duration":0.25411510467529297,"orig_bytes":77,"resp_bytes":295,"con
n_state":"SF","missed_bytes":0,"history":"ShADadFf","orig_pkts":6,"orig_ip_byte
s":397,"resp_pkts":4,"resp_ip_bytes":511}
```

Next we look at the dns log.

```
zeek@zeek:~/zeek-test/json$ cat dns.log

{"ts":1591367999.306059,"uid":"CMdzit1AMNsmfAIiQc","id.orig_h":"192.168.4.76","
id.orig_p":36844,"id.resp_h":"192.168.4.1","id.resp_p":53,"proto":"udp","trans_
id":8555,"query":"testmyids.com","qclass":1,"qclass_name":"C_INTERNET","qtype":
28,"qtype_name":"AAAA","rcode":0,"rcode_name":"NOERROR","AA":false,"TC":false,"
RD":true,"RA":false,"Z":0,"rejected":false}
{"ts":1591367999.305988,"uid":"CMdzit1AMNsmfAIiQc","id.orig_h":"192.168.4.76","
id.orig_p":36844,"id.resp_h":"192.168.4.1","id.resp_p":53,"proto":"udp","trans_
id":19671,"rtt":0.06685185432434082,"query":"testmyids.com","qclass":1,"qclass_
name":"C_INTERNET","qtype":1,"qtype_name":"A","rcode":0,"rcode_name":"NOERROR",
"AA":false,"TC":false,"RD":true,"RA":true,"Z":0,"answers":["31.3.245.133"],"TTL
s":[3600.0],"rejected":false}
```

Next we look at the files log.

```
zeek@zeek:~/zeek-test/json$ cat files.log

{"ts":1591367999.604,"fuid":"FEEsZS1w0Z0VJIb5x4","tx_hosts":["31.3.245.133"],"r
x_hosts":["192.168.4.76"],"conn_uids":["C5bLoe2Mvxqhawzqqd"],"source":"HTTP","d
epth":0,"analyzers":[],"mime_type":"text/plain","duration":0.0,"is_orig":false,
"seen_bytes":39,"total_bytes":39,"missing_bytes":0,"overflow_bytes":0,"timedout
":false}
```

Finally we look at the http log.

```
zeek@zeek:~/zeek-test/json$ cat http.log

{"ts":1591367999.512593,"uid":"C5bLoe2Mvxqhawzqqd","id.orig_h":"192.168.4.76","
id.orig_p":46378,"id.resp_h":"31.3.245.133","id.resp_p":80,"trans_depth":1,"met
hod":"GET","host":"testmyids.com","uri":"/","version":"1.1","user_agent":"curl/
7.47.0","request_body_len":0,"response_body_len":39,"status_code":200,"status_m
sg":"OK","tags":[],"resp_fuids":["FEEsZS1w0Z0VJIb5x4"],"resp_mime_types":["text
/plain"]}
```

Comparing the two log styles, we see strengths and weaknesses for each. For example, the TSV format shows the Zeek types associated with each entry, such as string, addr, port, and so on. The JSON format does not include that data. However, the JSON format associates each field “key” with a “value,” such as "id.orig_p":46378. This makes it easier for analysts and software to interpret the data, as the key is directly associated with the value that follows. For this reason, most developers and analysts have adopted the JSON output format for Zeek logs. That is the format we will use for the log analysis sections of the documentation.

Zeek JSON Format and Jq

Analysts sometimes choose to review JSON-formatted Zeek files using applications that recognize JSON format, such as jq. Jq is a JSON parser by Stephen Dolan, available at Github (https://stedolan.github.io/jq/). It may already be installed on your Unix-like system.

In the following example we process the dns.log file with the ‘.’ filter, which tells jq to simply output what it finds in the file. By default jq outputs JSON formatted data in its “pretty-print” style, which puts one key:value pair on each line as shown.

```
so16@so16:~/zeek-test/json$ jq . dns.log
```

```
{
"ts": 1591367999.306059,
"uid": "CMdzit1AMNsmfAIiQc",
"id.orig_h": "192.168.4.76",
"id.orig_p": 36844,
"id.resp_h": "192.168.4.1",
"id.resp_p": 53,
"proto": "udp",
"trans_id": 8555,
"query": "testmyids.com",
"qclass": 1,
"qclass_name": "C_INTERNET",
"qtype": 28,
"qtype_name": "AAAA",
"rcode": 0,
"rcode_name": "NOERROR",
"AA": false,
"TC": false,
"RD": true,
"RA": false,
"Z": 0,
"rejected": false
}
{
"ts": 1591367999.305988,
"uid": "CMdzit1AMNsmfAIiQc",
"id.orig_h": "192.168.4.76",
"id.orig_p": 36844,
"id.resp_h": "192.168.4.1",
"id.resp_p": 53,
"proto": "udp",
"trans_id": 19671,
"rtt": 0.06685185432434082,
"query": "testmyids.com",
"qclass": 1,
"qclass_name": "C_INTERNET",
"qtype": 1,
"qtype_name": "A",
"rcode": 0,
"rcode_name": "NOERROR",
"AA": false,
"TC": false,
"RD": true,
"RA": true,
"Z": 0,
"answers": [
"31.3.245.133"
],
"TTLs": [
3600
],
"rejected": false
}
```

We can tell jq to output what it sees in “compact” format using the -c switch.

```
so16@so16:~/zeek-test/json$ jq . -c dns.log

{"ts":1591367999.306059,"uid":"CMdzit1AMNsmfAIiQc","id.orig_h":"192.168.4.76","
id.orig_p":36844,"id.resp_h":"192.168.4.1","id.resp_p":53,"proto":"udp","trans_
id":8555,"query":"testmyids.com","qclass":1,"qclass_name":"C_INTERNET","qtype":
28,"qtype_name":"AAAA","rcode":0,"rcode_name":"NOERROR","AA":false,"TC":false,"
RD":true,"RA":false,"Z":0,"rejected":false}
{"ts":1591367999.305988,"uid":"CMdzit1AMNsmfAIiQc","id.orig_h":"192.168.4.76","
id.orig_p":36844,"id.resp_h":"192.168.4.1","id.resp_p":53,"proto":"udp","trans_
id":19671,"rtt":0.06685185432434082,"query":"testmyids.com","qclass":1,"qclass_
name":"C_INTERNET","qtype":1,"qtype_name":"A","rcode":0,"rcode_name":"NOERROR",
"AA":false,"TC":false,"RD":true,"RA":true,"Z":0,"answers":["31.3.245.133"],"TTL
s":[3600],"rejected":false}
```

The power of jq becomes evident when we decide we only want to see specific values. For example, the following tells jq to look at the dns.log and report the source IP of systems doingDNS queries, followed by the query, and any answer to the query.

```
so16@so16:~/zeek-test/json$ jq -c '["id.orig_h", ."query", ."answers"]' dns.log

["192.168.4.76","testmyids.com",null]
["192.168.4.76","testmyids.com",["31.3.245.133"]]
```

With this basic understanding of how to interact with Zeek logs, we can now turn to specific logs and interpret their values.

Conclusion

This section showed a sample of the sorts of logs that Zeek generates when processing a simple network trace. It explained the differences between logs in the traditional TSV format and the newer JSON format. It also demonstrated the use of a few simple command line tools to review Zeek logs in both formats.

Defenders have always sought the high ground in order to see farther and turn back attacks. Corelight delivers a commanding view of your network so you can outsmart and outlast adversaries. We capture, interpret, and connect the data that means everything to defenders.

info@corelight.com | 888-547-9497