# Explore signatures and logs with Suricata

## Project description

In this scenario, I monitored traffic on my employer's network. I configured Suricata and use it to trigger alerts.

Let’s define the files I’ll be working with in this lab activity:

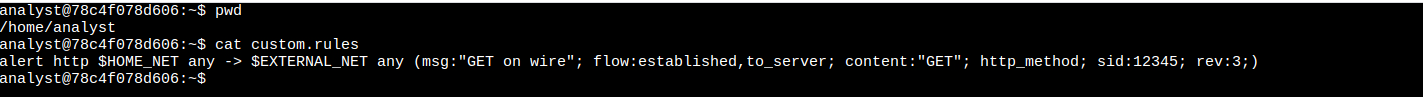
* The sample.pcap file is a packet capture file that contains an example of network traffic data, which I’ll use to test the Suricata rules. This will allow me to simulate and repeat the exercise of monitoring network traffic.
* The custom.rules file contains a custom rule when the lab activity starts. I’ll add rules to this file and run them against the network traffic data in the sample.pcap file.
* The fast.log file will contain the alerts that Suricata generates. The fast.log file is empty when the lab starts. Each time I test a rule, or set of rules, against the sample network traffic data, Suricata adds a new alert line to the fast.log file when all the conditions in any of the rules are met. The fast.log file can be located in the /var/log/suricata directory after Suricata runs.The fast.log file is considered to be a depreciated format and is not recommended for incident response or threat hunting tasks but can be used to perform quick checks or tasks related to quality assurance.
* The eve.json file is the main, standard, and default log for events generated by Suricata. It contains detailed information about alerts triggered, as well as other network telemetry events, in JSON format. The eve.json file is generated when Suricate runs, and can also be located in the /var/log/suricata directory.

## Examine a custom rule in Suricata

The /home/analyst directory contains a custom.rules file that defines the network traffic rules, which Suricata captures.

In this task, I explored the composition of the Suricata rule defined in the custom.rules file.

I use the cat command to display the rule in the custom.rules file:



This rule consists of three components: an action, a header, and rule options.

The **action** is the first part of the signature. It determines the action to take if all conditions are met.

Actions differ across network intrusion detection system (NIDS) rule languages, but some common actions are alert, drop, pass, and reject.

Using our example, the file contains a single alert as the action. The alert keyword instructs to alert on selected network traffic. The IDS will inspect the traffic packets and send out an alert in case it matches.

Note that the drop action also generates an alert, but it drops the traffic. A drop action only occurs when Suricata runs in IPS mode.

The pass action allows the traffic to pass through the network interface. The pass rule can be used to override other rules. An exception to a drop rule can be made with a pass rule. For example, the following rule has an identical signature to the previous example, except that it singles out a specific IP address to allow only traffic from that address to pass.

The reject action does not allow the traffic to pass. Instead, a TCP reset packet will be sent, and Suricata will drop the matching packet. A TCP reset packet tells computers to stop sending messages to each other.

The next part of the signature is the **header**. The header defines the signature’s network traffic, which includes attributes such as protocols, source and destination IP addresses, source and destination ports, and traffic direction.

The next field after the action keyword is the protocol field. In our example, the protocol is http, which determines that the rule applies only to HTTP traffic.

The parameters to the protocol http field are $HOME\_NET any -> $EXTERNAL\_NET any. The arrow indicates the direction of the traffic coming from the $HOME\_NET and going to the destination IP address $EXTERNAL\_NET.

$HOME\_NET is a Suricata variable defined in /etc/suricata/suricata.yaml that we can use in our rule definitions as a placeholder for our local or home network to identify traffic that connects to or from systems within our organization.

In this lab $HOME\_NET is defined as the 172.21.224.0/20 subnet.

The word any means that Suricata catches traffic from any port defined in the $HOME\_NET network.

So far, we know that this signature triggers an alert when it detects any http traffic leaving the home network and going to the external network.

The many available **rule options** allows to customize signatures with additional parameters. Configuring rule options helps narrow down network traffic so we can find exactly what we’re looking for. As in our example, rule options are typically enclosed in a pair of parentheses and separated by semicolons.

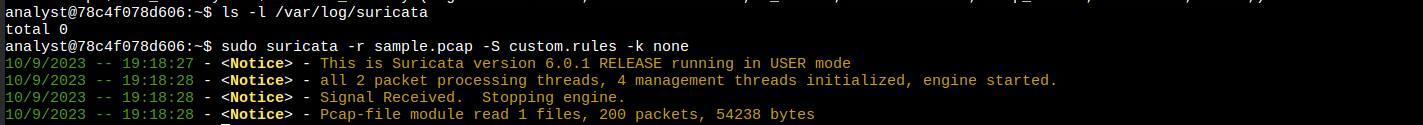
Let's further examine the rule options in our example:

* The msg: option provides the alert text. In this case, the alert will print out the text “GET on wire”, which specifies why the alert was triggered.
* The flow:established,to\_server option determines that packets from the client to the server should be matched. (In this instance, a server is defined as the device responding to the initial SYN packet with a SYN-ACK packet.)
* The content:"GET" option tells Suricata to look for the word GET in the content of the http.method portion of the packet.
* The sid:12345 (signature ID) option is a unique numerical value that identifies the rule.
* The rev:3 option indicates the signature's revision which is used to identify the signature's version. Here, the revision version is 3.

To summarize, this signature triggers an alert whenever Suricata observes the text GET as the HTTP method in an HTTP packet from the home network going to the external network.

Trigger a custom rule in Suricata

I triggered this rule and examine the alert logs that Suricata generates.

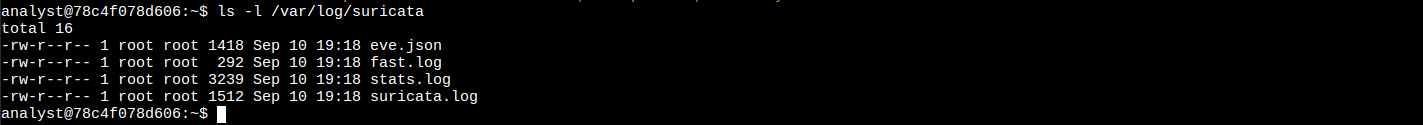
Note that before running Suricata, there are no files in the /var/log/suricata directory.

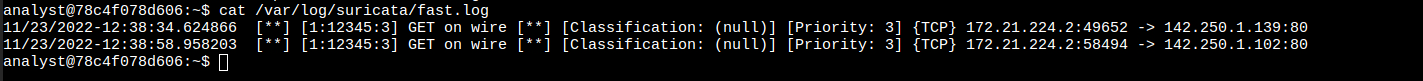
This command starts the Suricata application and processes the sample.pcap file using the rules in the custom.rules file. It returns an output stating how many packets were processed by Suricata.

The-r sample.pcap option specifies an input file to mimic network traffic. In this case, the sample.pcap file.

* The -S custom.rules option instructs Suricata to use the rules defined in the custom.rules file.
* The -k none option instructs Suricata to disable all checksum checks.
* checksums are a way to detect if a packet has been modified in transit.

Suricata adds a new alert line to the /var/log/suricata/fast.log file when all the conditions in any of the rules are met.

Note that after running Suricata, there are now four files in the /var/log/suricata directory, including the fast.log and eve.json files.

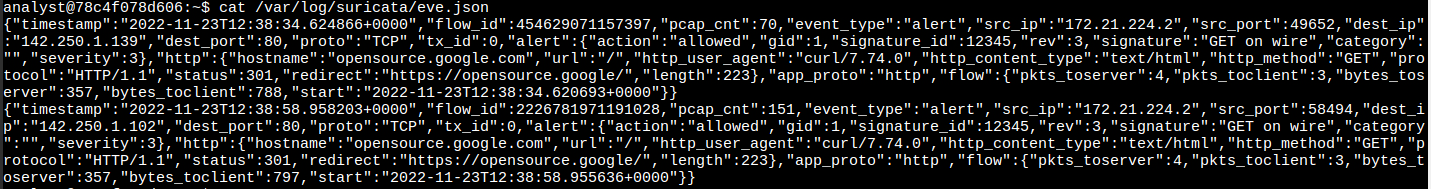
I used the cat command to display the fast.log file generated by Suricata

Each line or entry in the fast.log file corresponds to an alert generated by Suricata when it processes a packet that meets the conditions of an alert generating rule. Each alert line includes the message that identifies the rule that triggered the alert, as well as the source, destination, and direction of the traffic.

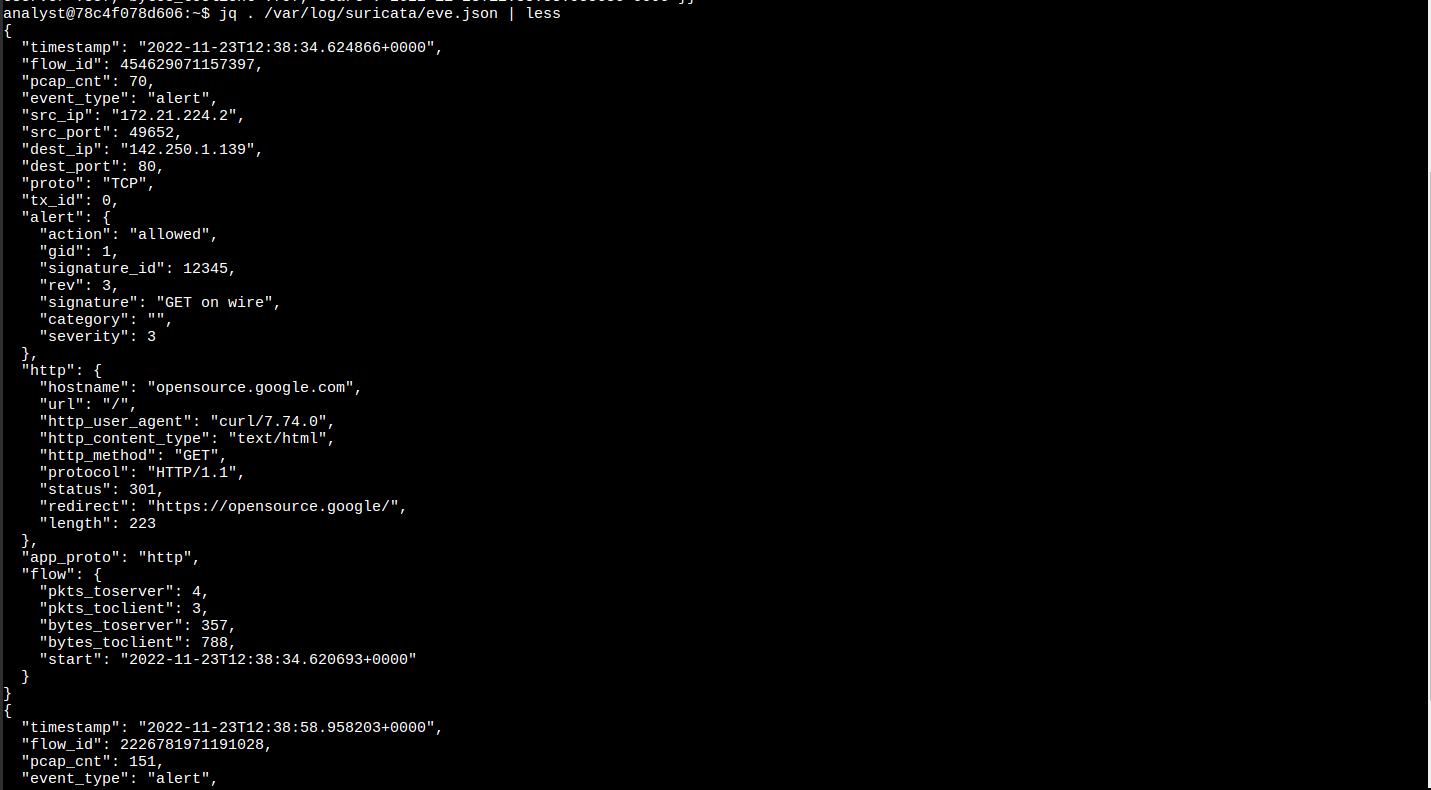
## Examine eve.json output

I examined the additional output that Suricata generates in the eve.json file.

As previously mentioned, this file is located in the /var/log/suricata/ directory. The eve.json file is the standard and main Suricata log file and contains a lot more data than the fast.log file. This data is stored in a JSON format, which makes it much more useful for analysis and processing by other applications.

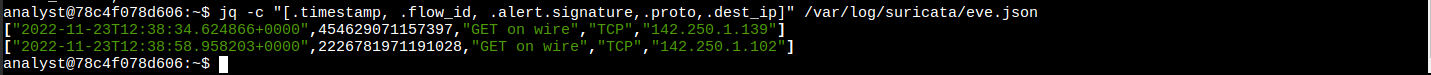
I used the cat command to display the entries in the eve.json file

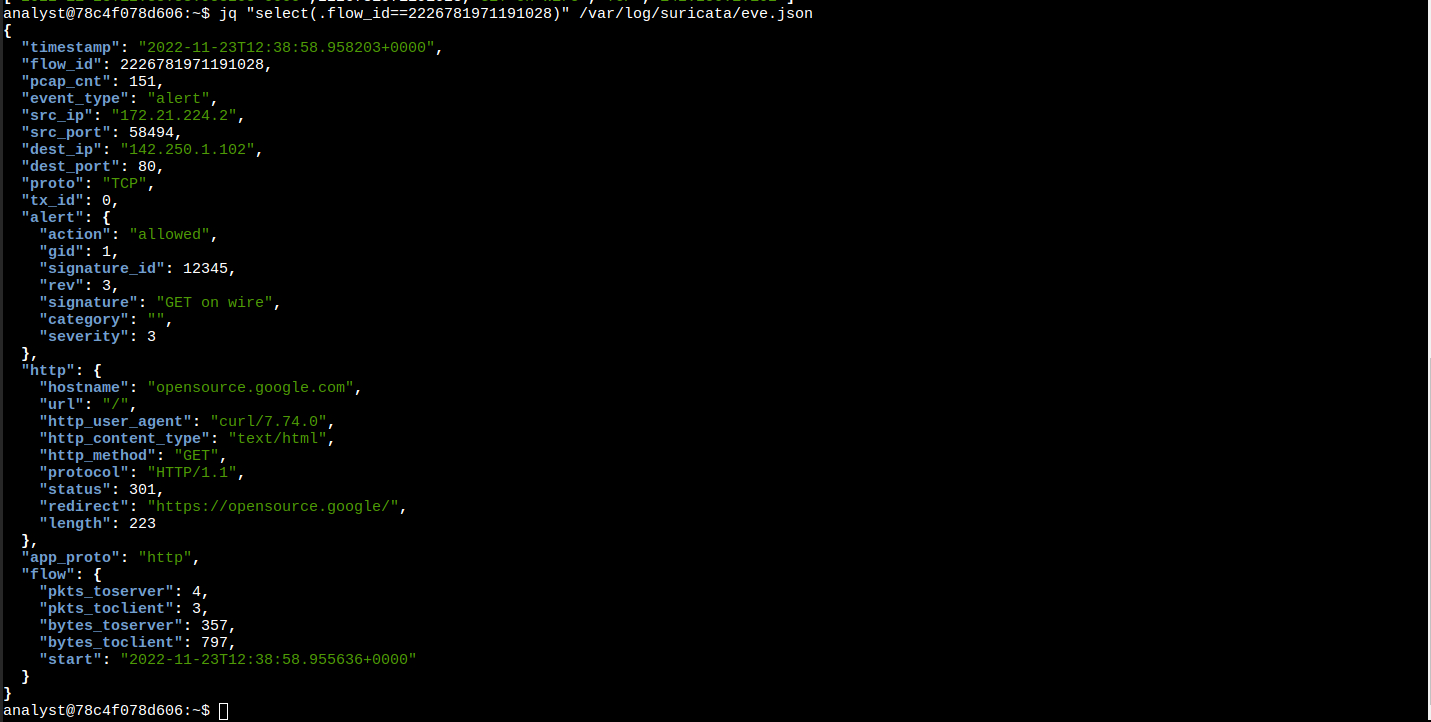
The output returns the raw content of the file.



I used the jq command to display the entries in an improved format

Pressed **Q** to exit the less command and to return to the command-line prompt.

Used the jq command to extract specific event data from the eve.json file:



I used the jq command to display all event logs related to a specific flow\_id from the eve.json file. The flow\_id value is a 16-digit number and will vary for each of the log entries.

## Summary

First, I explored custom rules in Suricata. Second, I ran Suricata with a custom rule in order to trigger it, and examine the output logs in the fast.log file. Finally, I examined the additional output that Suricata generates in the standard eve.json log file.