Examine alerts, logs, and rules with Suricata

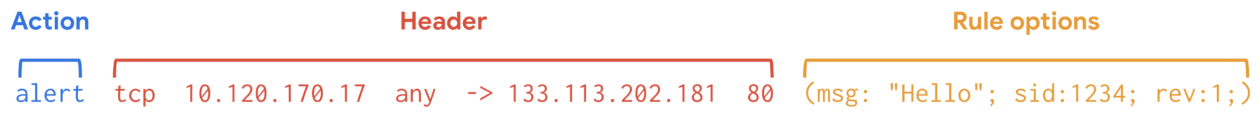
Overview

The Suricata tool monitors network interfaces and applies rules to the packets that pass through the interface. Suricata determines whether each packet should generate an alert and be dropped, rejected, or allowed to pass through the interface.

Suricata uses **signatures analysis**, which is a detection method used to find events of interest. Signatures consist of three components:

- **Action**: The first component of a signature. It describes the action to take if network or system activity matches the signature. Examples include: alert, pass, drop, or reject.
- **Header**: The header includes network traffic information like source and destination IP addresses, source and destination ports, protocol, and traffic direction.
- **Rule options:** The rule options provide you with different options to customize signatures.

Here's an example of a Suricata signature:



```
alert tcp 10.120.170.17 any -> 133.113.202.181 80 (msg: "Hello"; sid:1234; rev:1;)
```

Scenario

As a security analyst, I was tasked with monitoring the traffic on my employer's network. I configured Suricata and used it to trigger alerts.

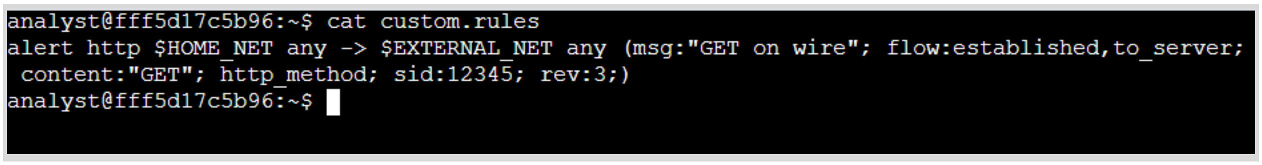
Task 1. Examine a custom rule in Suricata

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 used the `cat` command to display the rule in the `custom.rules` file:

```
cat custom.rules
```



```
analyst@fff5d17c5b96:~$ cat custom.rules
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"GET on wire"; flow:established,to_server;
 content:"GET"; http_method; sid:12345; rev:3;)
analyst@fff5d17c5b96:~$
```

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

The next field after the action keyword is the protocol field. In this case 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`.

The word `any` means that Suricata catches traffic from any port defined in the `$HOME_NET` network.

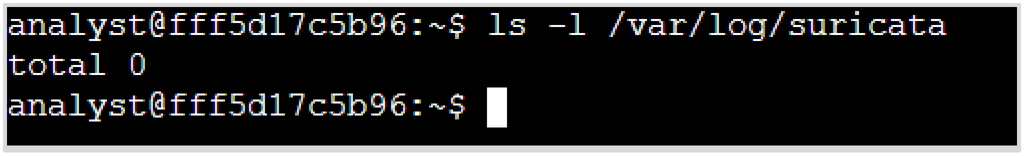
The next are the rule options used to customize signatures with additional parameters.

- The `msg`: The alert will print out the text GET on wire.
- The `flow:established, to_server`: Determines that packet from the client to the server should be matched (The handshakes: SYN-ACK packet).
- The content: `"GET"` tells Suricata to look for the word GET in the http.method of the packet.
- The `sid:12345`: Is a unique numerical value that identifies the rule.
- The `rev:3` indicates the signature's version.

Task 2. Trigger a custom rule in Suricata

I had to trigger this rule and examine the alert logs that Suricata generated:

```
ls -l /var/log/suricata
```

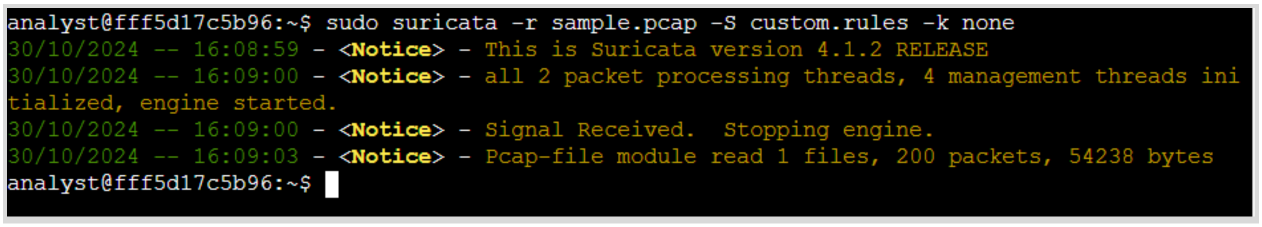


```
analyst@fff5d17c5b96:~$ ls -l /var/log/suricata
total 0
analyst@fff5d17c5b96:~$
```

I then ran Suricata using the `custom.rules` and `sample.pcap` files:

```
sudo suricata -r sample.pcap -S custom.rules -k none
```

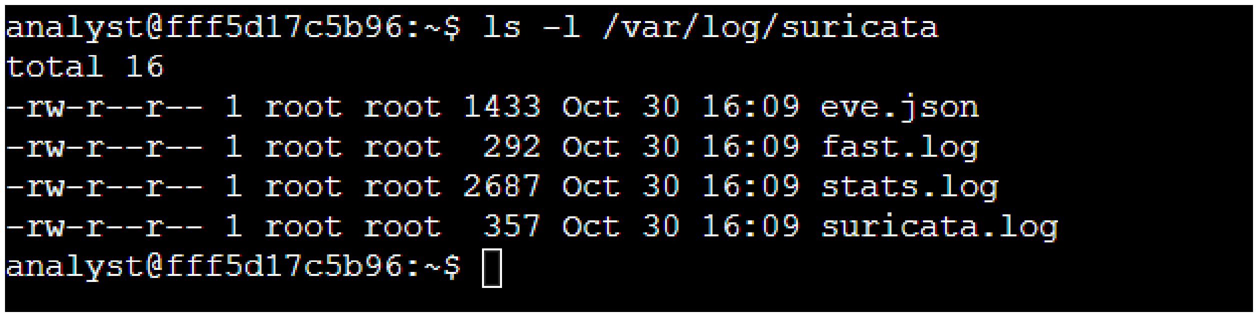
- The `-r sample.pcap` option specifies an input file to mimic network traffic. In this case, the sample.pcap file.
- The `-S custom.rules` option instructs Suricata to use the rules defined in the custom.rules file.
- The `-k none` option instructs Suricata to disable all checksum checks.



```
analyst@fff5d17c5b96:~$ sudo suricata -r sample.pcap -S custom.rules -k none
30/10/2024 -- 16:08:59 - <Notice> - This is Suricata version 4.1.2 RELEASE
30/10/2024 -- 16:09:00 - <Notice> - all 2 packet processing threads, 4 management threads ini
tialized, engine started.
30/10/2024 -- 16:09:00 - <Notice> - Signal Received.  Stopping engine.
30/10/2024 -- 16:09:03 - <Notice> - Pcap-file module read 1 files, 200 packets, 54238 bytes
analyst@fff5d17c5b96:~$
```

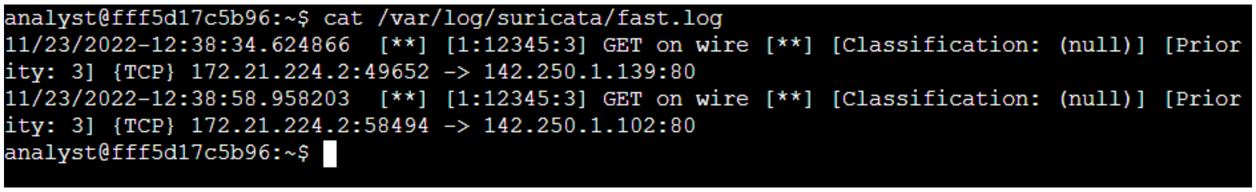
This command started the Suricata application and processed the sample.pcap file using the rules in the custom.rules file. It returned an output stating how many packets were processed by Suricata.

I listed the files in the `/var/log/suricata` folder again:



```
analyst@fff5d17c5b96:~$ ls -l /var/log/suricata
total 16
-rw-r--r-- 1 root root 1433 Oct 30 16:09 eve.json
-rw-r--r-- 1 root root  292 Oct 30 16:09 fast.log
-rw-r--r-- 1 root root 2687 Oct 30 16:09 stats.log
-rw-r--r-- 1 root root  357 Oct 30 16:09 suricata.log
analyst@fff5d17c5b96:~$
```

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



```
analyst@fff5d17c5b96:~$ cat /var/log/suricata/fast.log
11/23/2022-12:38:34.624866  [**] [1:12345:3] GET on wire [**] [Classification: (null)] [Prior
ity: 3] {TCP} 172.21.224.2:49652 -> 142.250.1.139:80
11/23/2022-12:38:58.958203  [**] [1:12345:3] GET on wire [**] [Classification: (null)] [Prior
ity: 3] {TCP} 172.21.224.2:58494 -> 142.250.1.102:80
analyst@fff5d17c5b96:~$
```

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.

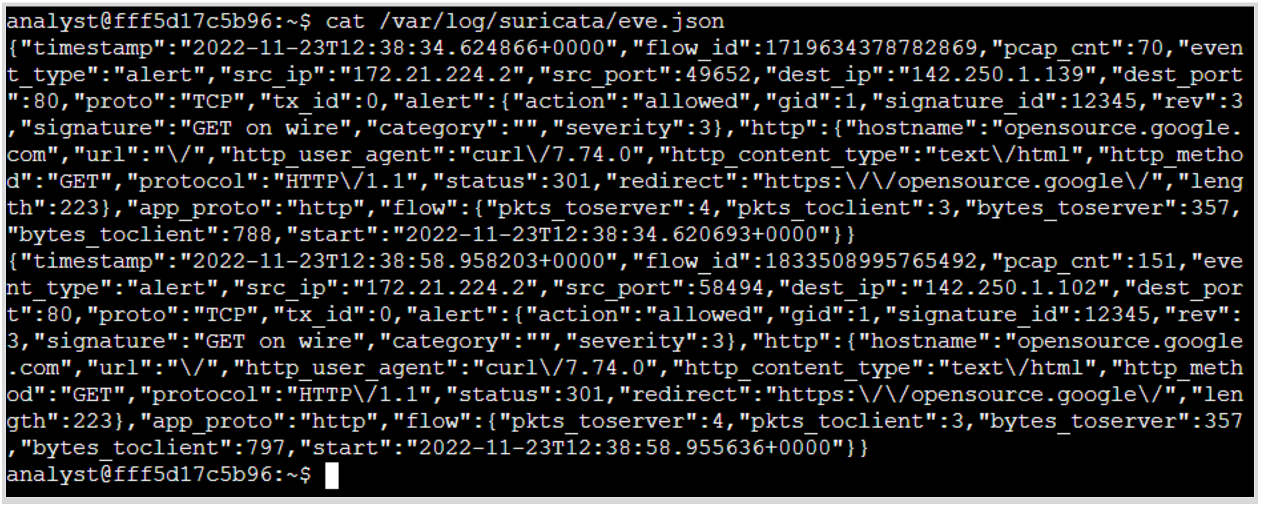
Task 3. Examine eve.json output

I had to examine the additional output that Suricata generated in the eve.json file.

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 next cat command to display the entries in the eve.json file:

`cat /var/log/suricata/eve.json`



```
analyst@fff5d17c5b96:~$ cat /var/log/suricata/eve.json
{"timestamp":"2022-11-23T12:38:34.624866+0000","flow_id":1719634378782869,"pcap_cnt":70,"even
t_type":"alert","src_ip":"172.21.224.2","src_port":49652,"dest_ip":"142.250.1.139","dest_port
":80,"proto":"TCP","tx_id":0,"alert":{"action":"allowed","gid":1,"signature_id":12345,"rev":3
,"signature":"GET on wire","category":"","severity":3},"http":{"hostname":"opensource.google.
com","url":"\/","http_user_agent":"curl\/7.74.0","http_content_type":"text\/html","http_metho
d":"GET","protocol":"HTTP\/1.1","status":301,"redirect":"https:\/\/opensource.google\/","leng
th":223},"app_proto":"http","flow":{"pkts_toserver":4,"pkts_toclient":3,"bytes_toserver":357,
"bytes_toclient":788,"start":"2022-11-23T12:38:34.620693+0000"}}
{"timestamp":"2022-11-23T12:38:58.958203+0000","flow_id":1833508995765492,"pcap_cnt":151,"eve
nt_type":"alert","src_ip":"172.21.224.2","src_port":58494,"dest_ip":"142.250.1.102","dest_por
t":80,"proto":"TCP","tx_id":0,"alert":{"action":"allowed","gid":1,"signature_id":12345,"rev":
3,"signature":"GET on wire","category":"","severity":3},"http":{"hostname":"opensource.google
.com","url":"\/","http_user_agent":"curl\/7.74.0","http_content_type":"text\/html","http_meth
od":"GET","protocol":"HTTP\/1.1","status":301,"redirect":"https:\/\/opensource.google\/","len
gth":223},"app_proto":"http","flow":{"pkts_toserver":4,"pkts_toclient":3,"bytes_toserver":357
,"bytes_toclient":797,"start":"2022-11-23T12:38:58.955636+0000"}}
analyst@fff5d17c5b96:~$
```

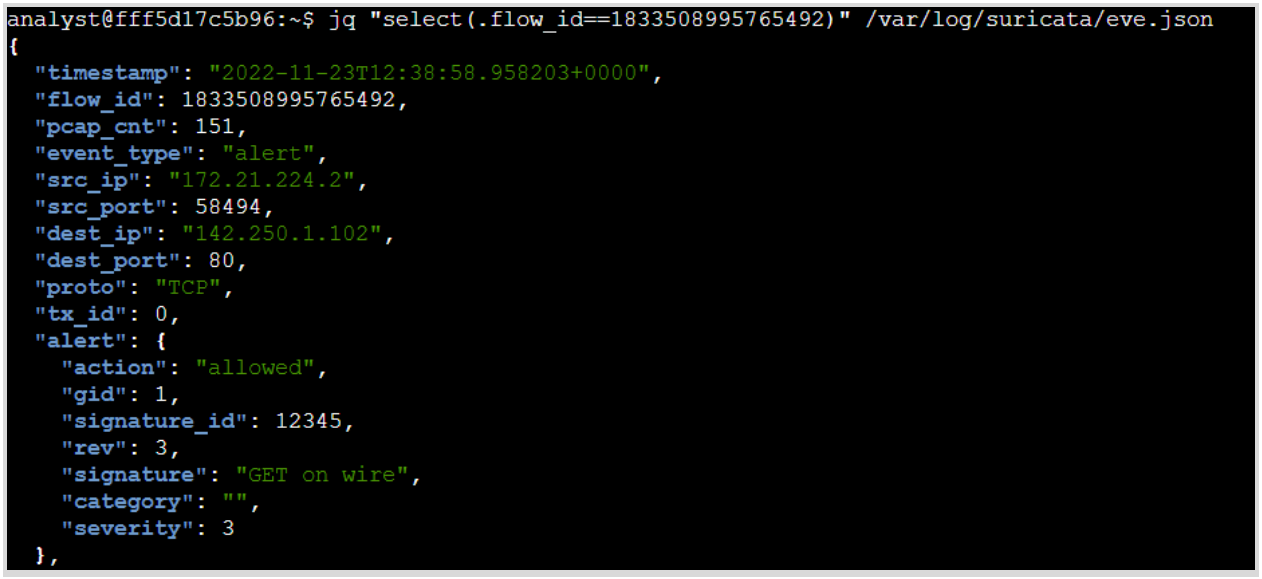
I used the `jq` command to extract specific event data from the `eve.json` file:

```
jq -c "[.timestamp,.flow_id,.alert.signature,.proto,.dest_ip]"
/var/log/suricata/eve.json
```

The fields selected are the timestamp (`.timestamp`), the flow id (`.flow_id`), the alert signature or msg (`.alert.signature`), the protocol (`.proto`), and the destination IP address (`.dest_ip`).

Next I used the `jq` command to display all event logs related to a specific 16-digit flow_id from the `eve.json` file.

```
jq "select(.flow_id==1833508995765492)" /var/log/suricata/eve.json
```



```
analyst@fff5d17c5b96:~$ jq "select(.flow_id==1833508995765492)" /var/log/suricata/eve.json
{
  "timestamp": "2022-11-23T12:38:58.958203+0000",
  "flow_id": 1833508995765492,
  "pcap_cnt": 151,
  "event_type": "alert",
  "src_ip": "172.21.224.2",
  "src_port": 58494,
  "dest_ip": "142.250.1.102",
  "dest_port": 80,
  "proto": "TCP",
  "tx_id": 0,
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 12345,
    "rev": 3,
    "signature": "GET on wire",
    "category": "",
    "severity": 3
  },
```

Summary

I created custom rules and ran them in Suricata, monitored traffic captured in a packet capture file, and examined the `fast.log` and `eve.json` output.