Scenario

In this scenario, you're a security analyst who must monitor traffic on your employer's network. You'll be required to configure Suricata and use it to trigger alerts.

Here's how you'll do this task: **First**, you'll explore custom rules in Suricata. **Second**, you'll run Suricata with a custom rule in order to trigger it, and examine the output logs in the `fast.log` file. **Finally**, you'll examine the additional output that Suricata generates in the standard `eve.json` log file.

For the purposes of the tests you'll run in this lab activity, you've been supplied with a `sample.pcap` file and a `custom.rules` file. These reside in your home folder.

Let's define the files you'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 you'll use to test the Suricata rules. This will allow you to simulate and repeat the exercise of monitoring network traffic.
- The `custom.rules` file contains a custom rule when the lab activity starts. You'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 you 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.

When you create a new rule, you'll need to test the rule to confirm whether or not it worked as expected. You can use the `fast.log` file to quickly compare the number of alerts generated each time you run Suricata to test a signature against the `sample.pcap` file.

It’s time to get started.

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, you’ll explore the composition of the Suricata rule defined in the `custom.rules` file.

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

```
analyst@acca80fdabf1:~$ cat custom.rules
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"GET on wire"; flow:established,to_ser
ver; content:"GET"; http_method; sid:12345; rev:3;)
analyst@acca80fdabf1:~$
```

Action = Alert: Instructs to alert on selected network traffic. The IDS will inspect the traffic packets and send out an alert in case it matches.

Header = HTTP: The rule applies to only 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 you can use in your rule definitions as a placeholder for your local or home network to identify traffic that connects to or from systems within your organization.
In this lab $HOME_NET is defined as the 172.21.224.0/20 subnet.

Rule Options = Rule: 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: Unique numerical value that identify the rule.
- The rev:3 indicates the signature's version which is used to identify the signature's version.

Task 2. Trigger a custom rule in Suricata

Now that you are familiar with the composition of the custom Suricata rule, you must trigger this rule and examine the alert logs that Suricata generates.

1. List the files in the `/var/log/suricata` folder:

```
analyst@acca80fdabf1:~$ cat custom.rules
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"GET on wire"; flow:established,to_ser
ver; content:"GET"; http_method; sid:12345; rev:3;)
analyst@acca80fdabf1:~$ ls -l /var/log/suricata
total 0
analyst@acca80fdabf1:~$
```

2. Run `suricata` using the `custom.rules` and `sample.pcap` files:

```
analyst@acca80fdabf1:~$ ls -l /var/log/suricata
total 0
analyst@acca80fdabf1:~$ sudo suricata -r sample.pcap -S custom.rules -k none
21/8/2024 -- 19:24:06 - <Notice> - This is Suricata version 4.1.2 RELEASE
21/8/2024 -- 19:24:07 - <Notice> - all 2 packet processing threads, 4 management threads
initialized, engine started.
21/8/2024 -- 19:24:07 - <Notice> - Signal Received.  Stopping engine.
21/8/2024 -- 19:24:09 - <Notice> - Pcap-file module read 1 files, 200 packets, 54238 byte
s
analyst@acca80fdabf1:~$
```

- 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.

3. List the files in the `/var/log/suricata` folder again:

```
21/8/2024 -- 19:24:07 - <Notice> - all 2 packet processing threads, 4 management threads
initialized, engine started.
21/8/2024 -- 19:24:07 - <Notice> - Signal Received.  Stopping engine.
21/8/2024 -- 19:24:09 - <Notice> - Pcap-file module read 1 files, 200 packets, 54238 byte
s
analyst@acca80fdabf1:~$ ls -l /var/log/suricata
total 16
-rw-r--r-- 1 root root 1433 Aug 21 19:24 eve.json
-rw-r--r-- 1 root root  292 Aug 21 19:24 fast.log
-rw-r--r-- 1 root root 2686 Aug 21 19:24 stats.log
-rw-r--r-- 1 root root  353 Aug 21 19:24 suricata.log
analyst@acca80fdabf1:~$
```

4. Use 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.

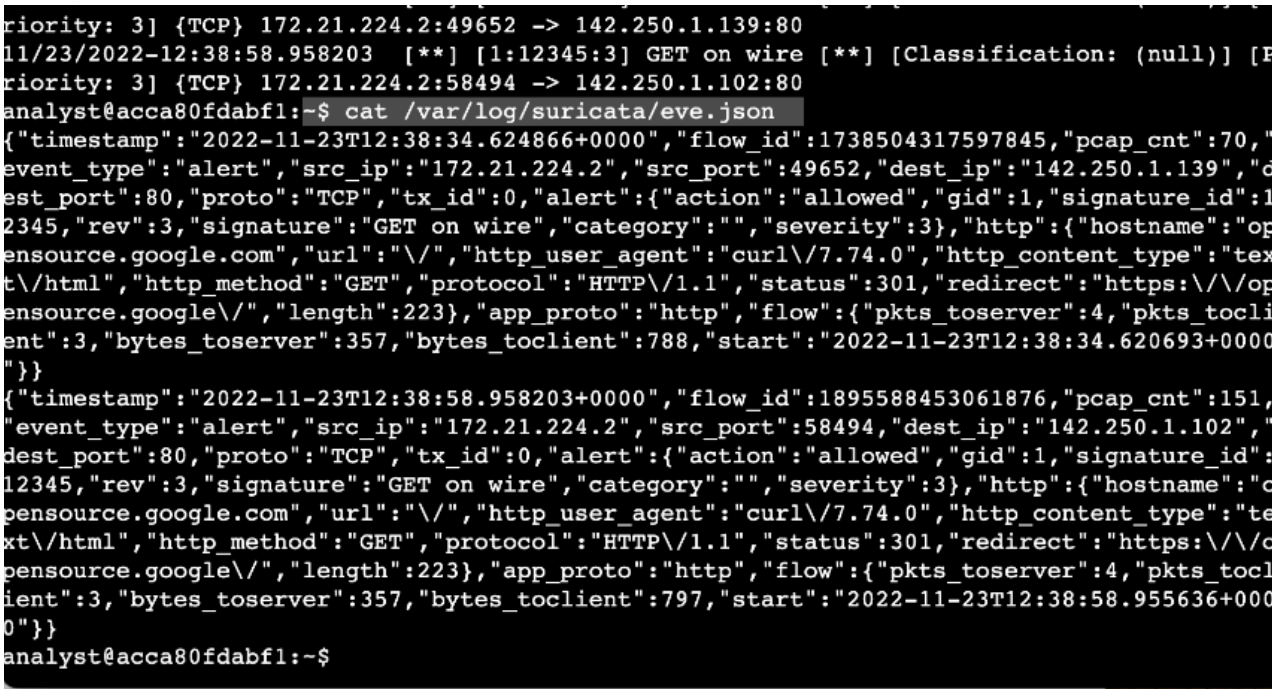
Task 3. Examine eve.json output

In this task, you must examine 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.

1. Use the `cat` command to display the entries in the `eve.json` file:



2. Use the `jq` command to display the entries in an improved format:

```
0"}}
analyst@acca80fdabf1:~$ jq . /var/log/suricata/eve.json | less
{
  "timestamp": "2022-11-23T12:38:34.624866+0000",
  "flow_id": 1738504317597845,
  "pcap_cnt": 70,
  "event_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_method": "GET",
    "protocol": "HTTP/1.1",
    "status": 301,
    "redirect": "https://opensource.google/",
    "length": 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"
  }
}
```

3. Press **Q** to exit the `less` command and to return to the command-line prompt.

4. Use the `jq` command to extract specific event data from the `eve.json` file:

```
    "bytes_toclient": 788,
    "start": "2022-11-23T12:38:34.620693+0000"
  }
}
{
  "timestamp": "2022-11-23T12:38:58.958203+0000",
analyst@acca80fdabf1:~$ jq -c "[.timestamp,.flow_id,.alert.signature,.proto,.dest_ip]" /var/log/suricata
/eve.json
["2022-11-23T12:38:34.624866+0000",1738504317597845,"GET on wire","TCP","142.250.1.139"]
["2022-11-23T12:38:58.958203+0000",1895588453061876,"GET on wire","TCP","142.250.1.102"]
analyst@acca80fdabf1:~$
```

5. Use 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. Replace `X` with any of the `flow_id` values returned by the previous query: