Scénario

Dans ce scénario, vous êtes un analyste de sécurité et devez surveiller le trafic sur le réseau de votre employeur. Vous devrez configurer Suricata et l'utiliser pour déclencher des alertes.

Voici comment vous allez y parvenir : **pour commencer**, vous allez découvrir les règles personnalisées dans Suricata. **Ensuite**, vous allez exécuter et déclencher une règle personnalisée avec Suricata, puis examiner les journaux de sortie dans le fichier `fast.log`. **Pour terminer**, vous allez examiner la sortie supplémentaire générée par Suricata dans le fichier journal `eve.json` standard.

Pour les tests que vous devrez réaliser dans cet atelier, nous vous avons fourni un fichier `sample.pcap` et un fichier `custom.rules`. Vous les trouverez dans votre dossier de base.

Commençons par définir les fichiers avec lesquels vous allez travailler dans cet atelier :

- Le fichier `sample.pcap` est un fichier de capture de paquets qui contient un exemple de données de trafic réseau, dont vous vous servirez pour tester les règles de Suricata. Cela vous permettra de simuler et de reproduire l'exercice qui consiste à surveiller le trafic réseau.

- Le fichier `custom.rules` contient une règle personnalisée au début de l'atelier. Vous y ajouterez des règles et les exécuterez en fonction des données du trafic

réseau contenues dans le fichier `sample.pcap`.

- Le fichier `fast.log` contiendra les alertes générées par Suricata. Au début de l'atelier, ce fichier est vide. Chaque fois que vous testez une règle ou un ensemble de règles par rapport à l'exemple de données du trafic réseau, Suricata ajoute une nouvelle ligne d'alerte au fichier `fast.log` lorsque toutes les conditions de l'une des règles sont satisfaites. Vous trouverez le fichier `fast.log` dans le répertoire `/var/log/suricata` une fois Suricata lancé. Le format du fichier `fast.log` est considéré comme obsolète et n'est pas recommandé pour les tâches de réponse aux incidents ou de recherche de menaces. Vous pouvez toutefois l'utiliser pour effectuer des vérifications rapides ou des tâches liées au contrôle qualité.

- Le fichier `eve.json` est le journal principal standard par défaut pour les événements générés par Suricata. Il contient des informations détaillées sur les alertes déclenchées, ainsi que sur d'autres événements de télémétrie concernant le réseau, au format JSON. Le fichier `eve.json` est généré lorsque Suricata s'exécute et se trouve également dans le répertoire `/var/log/suricata`.

Tâche 1 : Examiner une règle personnalisée dans Suricata

Le répertoire `/home/analyst` contient un fichier `custom.rules` qui définit les règles du trafic réseau que Suricata capture.

Dans cette tâche, vous allez découvrir la composition de la règle Suricata définie dans le fichier `custom.rules`.

```
analyst@b6a5fcc22fd3:~$ cat custom.rules
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"GET on wire"; flow:established,to_server; content:
"GET"; http_method; sid:12345; rev:3;)
analyst@b6a5fcc22fd3:~$
```

La commande renvoie la règle sous la forme d'une sortie dans le shell.
Cette règle se compose de trois éléments : une **action**, un **en-tête** et des **options de règle**.
L'**action** est la première partie de la signature. Elle détermine l'action à réaliser lorsque toutes les conditions sont remplies.

L’action

Les actions varient en fonction du langage des règles du système de détection des intrusions du réseau, mais `alert`, `drop`, `pass` et `reject` sont des actions courantes.

Dans notre exemple, le fichier ne contient qu'une seule action : `alert`. Le mot clé `alert` signifie qu'une alerte doit être émise pour le trafic réseau sélectionné. L'IDS inspectera les paquets de trafic et émettra une alerte si les conditions sont remplies.

Notez que l'action `drop` génère également une alerte, mais abandonne le trafic. L'action `drop` n'est effectuée que lorsque Suricata s'exécute en mode IPS.

L'action `pass` autorise le trafic à circuler via l'interface réseau. Elle peut être utilisée pour ignorer d'autres règles. Une règle pass peut permettre de créer une exception à la règle drop. Par exemple, la signature de la règle suivante est identique à l'exemple précédent, si ce n'est qu'elle spécifie la seule adresse IP depuis laquelle le trafic peut être transmis :

L'action `reject` ne permet pas au trafic d'être transmis. Au lieu de cela, un paquet de réinitialisation TCP est envoyé et Suricata abandonne le paquet correspondant. Un paquet de réinitialisation TCP indique aux ordinateurs d'arrêter de s'envoyer des messages.

L’en-tête

L'élément suivant de la signature est l'**en-tête**. L'en-tête définit le trafic réseau de la signature, qui inclut des attributs comme les protocoles, les adresses IP sources et de destination, les ports sources et de destination ainsi que le sens du trafic.

Après le mot clé de l'action vient le champ du protocole. Dans cet exemple, le protocole `http` indique que la règle ne s'applique qu'au trafic HTTP.

Les paramètres du champ de protocole `http` sont `$HOME_NET any -> $EXTERNAL_NET any`. La flèche indique le sens du trafic, qui provient de l'adresse `$HOME_NET` et se dirige vers l'adresse IP de destination `$EXTERNAL_NET`.

`$HOME_NET` est une variable Suricata définie dans `/etc/suricata/suricata.yaml`. Vous pouvez l'utiliser dans les définitions de vos règles comme espace réservé pour votre réseau local ou domestique, afin d'identifier le trafic connecté aux systèmes au sein de votre organisation.

Dans cet atelier, `$HOME_NET` est défini comme le sous-réseau 172.21.224.0/20.

Le mot `any` signifie que Suricata capture le trafic provenant de n'importe quel port défini dans le réseau `$HOME_NET`.

***Remarque** : Le symbole `$` marque le début d'une variable. Les variables servent d'espaces réservés pour stocker des valeurs.*

Pour l'instant, nous savons que cette signature déclenche une alerte lorsqu'elle détecte un trafic HTTP provenant du réseau domestique et se dirigeant vers le réseau externe.

Options des règles

Les nombreuses **options disponibles pour les règles** vous permettent de personnaliser les signatures à l'aide de paramètres supplémentaires. Configurer les options des règles vous permet de mieux définir le trafic réseau pour que vous trouviez exactement ce que vous cherchez. Comme dans cet exemple, les options des règles sont généralement placées entre parenthèses et séparées par des points-virgules.

Examinons plus en détail les options de notre exemple :

- L'option `msg:` permet d'indiquer le texte de l'alerte. Dans ce cas, l'alerte affiche le texte `“GET on wire”`, qui indique le motif de déclenchement de l'alerte.
- L'option `flow:established,to_server` indique que les paquets du client vers le serveur doivent être mis en correspondance. Dans le cas présent, un serveur est défini comme étant l'appareil qui répond au paquet SYN initial par un paquet SYN-ACK.
- L'option `content:"GET"` indique à Suricata de rechercher le mot `GET` dans le contenu de la partie `http.method` du paquet.
- L'option `sid:12345` (identifiant de la signature) est une valeur numérique unique qui sert à identifier la règle.
- L'option `rev:3` indique la révision de la signature qui est utilisée pour identifier la version de la signature. Ici, il s'agit de la révision 3.

En résumé, cette signature déclenche une alerte chaque fois que Suricata détecte le texte `GET` comme méthode HTTP dans un paquet HTTP provenant du réseau domestique et se dirigeant vers le réseau externe.

Tâche 2 : Déclencher une règle personnalisée dans Suricata

Maintenant que vous connaissez la composition d'une règle Suricata personnalisée, vous devez déclencher cette règle et examiner les journaux d'alerte générés par Suricata.

1. Affichez les fichiers contenus dans le dossier `/var/log/suricata` :

La commande executée est : **ls -l /var/log/suricata**

```
analyst@b6a5fcc22fd3:~$ ls -l /var/log/suricata
total 0
analyst@b6a5fcc22fd3:~$
```

Notez qu'avant l'exécution de Suricata, le répertoire `/var/log/suricata` est vide.

2. Exécutez `suricata` à l'aide des fichiers `custom.rules` et `sample.pcap` :

La commande exécutée est : sudo suricata -r sample.pcap -S custom.rules -k none

Cette commande lance l'application Suricata et traite le fichier `sample.pcap` à l'aide des règles du fichier `custom.rules`. La sortie renvoyée indique le nombre de paquets traités par Suricata.

Examinons maintenant les options de la commande :

- L'option `-r sample.pcap` spécifie un fichier d'entrée pour simuler un trafic réseau. Il s'agit, dans ce cas, du fichier `sample.pcap`.
- L'option `-S custom.rules` indique à Suricata d'utiliser les règles définies dans le fichier `custom.rules`.
- L'option `-k none` indique à Suricata de désactiver toutes les vérifications de somme de contrôle.

Pour mémoire, les sommes de contrôle permettent de détecter si un paquet a été modifié pendant le transit. Comme vous utilisez un trafic réseau provenant d'un exemple de fichier de capture de paquets, vous n'avez pas besoin que Suricata vérifie l'intégrité de la somme de contrôle.

```
analyst@b6a5fcc22fd3:~$ sudo suricata -r sample.pcap -S custom.rules -k none
18/9/2025 -- 20:42:43 - <Notice> - This is Suricata version 6.0.1 RELEASE running in USER mode
18/9/2025 -- 20:42:44 - <Notice> - all 2 packet processing threads, 4 management threads initialized,
engine started.
18/9/2025 -- 20:42:44 - <Notice> - Signal Received.  Stopping engine.
18/9/2025 -- 20:42:44 - <Notice> - Pcap-file module read 1 files, 200 packets, 54238 bytes
analyst@b6a5fcc22fd3:~$
```

3. Affichez à nouveau les fichiers contenus dans le dossier `/var/log/suricata` :

```
analyst@b6a5fcc22fd3:~$ ls -l /var/log/suricata
total 16
-rw-r--r-- 1 root root 1418 Sep 18 20:42 eve.json
-rw-r--r-- 1 root root  292 Sep 18 20:42 fast.log
-rw-r--r-- 1 root root 2846 Sep 18 20:42 stats.log
-rw-r--r-- 1 root root 1512 Sep 18 20:42 suricata.log
analyst@b6a5fcc22fd3:~$
```

4. Utilisez la commande `cat` pour afficher le fichier `fast.log` généré par Suricata :

La commande exécutée est: cat /var/log/suricata/fast.log

Le journal renvoyé contient des entrées d'alerte :

```
analyst@b6a5fcc22fd3:~$ cat /var/log/suricata/fast.log
11/23/2022-12:38:34.624866  [**] [1:12345:3] GET on wire [**] [Classification: (null)] [Priority: 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)] [Priority: 3] {
TCP} 172.21.224.2:58494 -> 142.250.1.102:80
analyst@b6a5fcc22fd3:~$
```

Chaque ligne ou entrée du fichier `fast.log` correspond à une alerte générée par Suricata quand il traite un paquet qui remplit les conditions d'une règle de génération d'alerte. Chaque ligne d'alerte inclut le message qui identifie la règle qui a déclenché l'alerte, ainsi que la source, la destination et le sens du trafic.

Tâche 3 : Examiner la sortie du fichier eve.json

Dans cette tâche, vous allez examiner la sortie supplémentaire que Suricata génère dans le fichier `eve.json`.

Comme indiqué précédemment, ce fichier se trouve dans le répertoire `/var/log/suricata/`.

Le fichier `eve.json` est le fichier journal standard principal de Suricata. Il contient beaucoup plus de données que le fichier `fast.log`. Ces données sont stockées au format JSON, ce qui est bien plus pratique pour les analyser et les traiter avec d'autres applications.

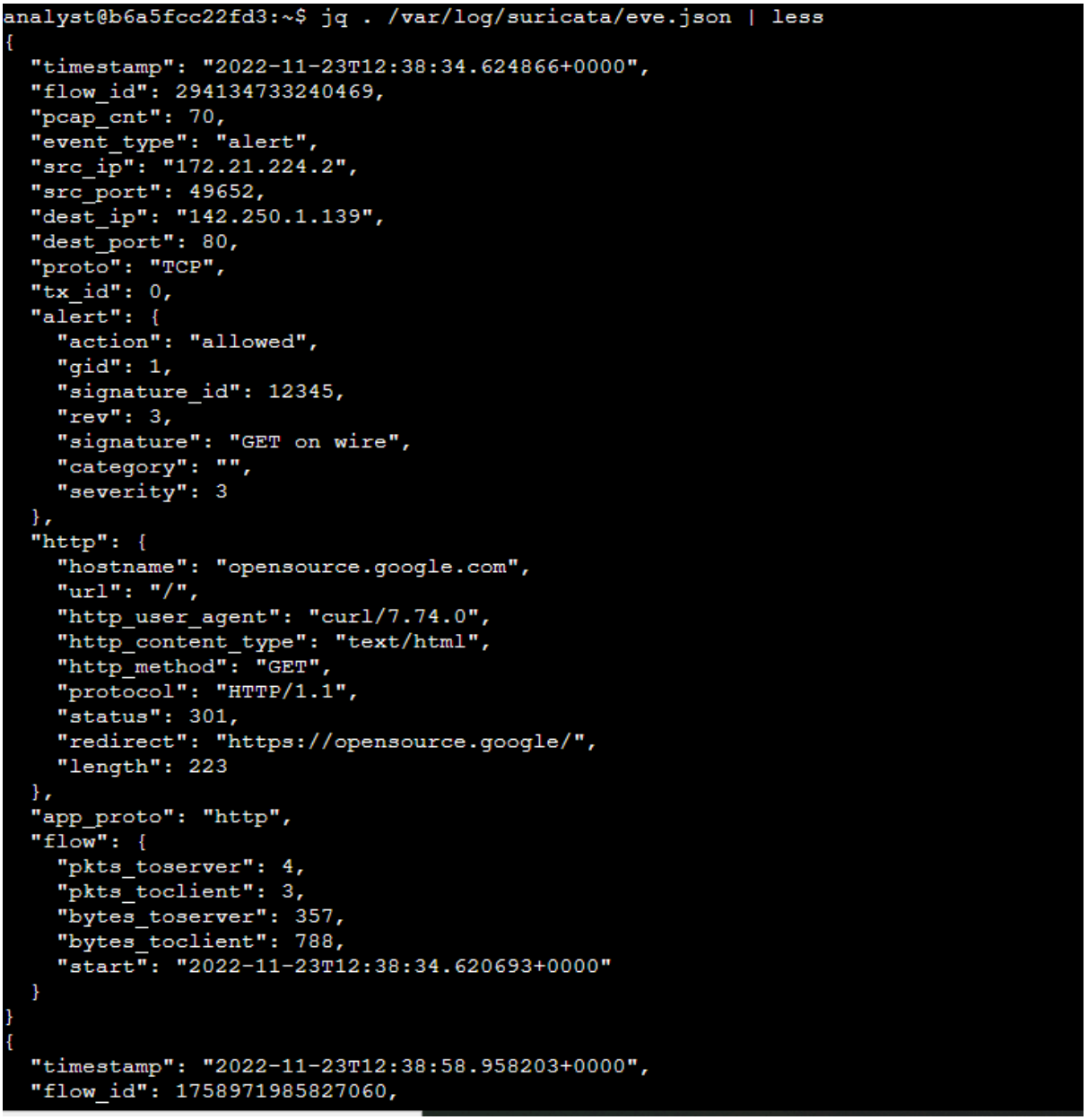
1. Utilisez la commande `cat` pour afficher les entrées du fichier `eve.json`

La commande exécutée est : cat /var/log/suricata/eve.json

Le contenu brut du fichier est renvoyé. Comme vous pouvez le constater, un grand nombre de données est renvoyé, mais le format ne permet pas de les comprendre aisément.

2. Utilisez la commande `jq` pour afficher les entrées dans un meilleur format

La commande exécutée est jq . /var/log/suricata/eve.json | less



Notez que la sortie est bien plus facile à lire qu'avec le résultat de la commande `cat`.

Remarque : L'outil `jq` est très pratique pour traiter les données JSON.

3. Utilisez la commande `jq` pour extraire des données d'événement spécifiques du fichier `eve.json` :

La commande utilisée est : jq -c "[.timestamp,.flow_id,.alert.signature,.proto,.dest_ip]" /var/log/suricata/eve.json

Remarque : La commande `jq` ci-dessus extrait de la charge utile JSON les champs spécifiés dans la liste entre crochets. Les champs sélectionnés sont le code temporel

(`.timestamp`), le flux (`.flow_id`), la signature ou le message d'alerte (`.alert.signature`), le protocole (`.proto`), et l'adresse IP de destination (`.dest_ip`).

4. Utilisez la commande `jq` pour afficher tous les journaux d'événements associés à un `flow_id` spécifique du fichier `eve.json`. La valeur de `flow_id` est un nombre à 16 chiffres. Elle varie pour chaque entrée de journal. Remplacez `X` par n'importe quelle valeur de `flow_id` renvoyée par la requête précédente :

La commande exécutée est : jq "select(.flow_id==294134733240469)" /var/log/suricata/eve.json

```
analyst@b6a5fcc22fd3:~$ jq "select(.flow_id==294134733240469)" /var/log/suricata/eve.json
{
  "timestamp": "2022-11-23T12:38:34.624866+0000",
  "flow_id": 294134733240469,
  "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"
  }
}
```

***Remarque** : Un flux réseau désigne une séquence de paquets entre une source et une destination qui partagent certaines caractéristiques, comme des adresses IP, des protocoles, etc. En cybersécurité, les flux de trafic réseau aident les analystes à comprendre le comportement du trafic réseau, et ainsi à identifier et à analyser les menaces. Suricata attribue un `flow_id` unique à chaque flux réseau. Tous les journaux d'un flux réseau partagent le même `flow_id`. Le champ `flow_id` est dès lors bien pratique pour corréler le trafic réseau appartenant aux mêmes flux réseau.*