

**Mise en place**

# Introduction

Suricata est un outil de surveillance de la sécurité du réseau (NSM) qui utilise des ensembles de signatures créées par la communauté et définies par l'utilisateur (également appelées règles) afin d’examiner et traiter le trafic réseau.

Suricata peut …

* générer des événements de journal ;
* déclencher des alertes   
  et
* interrompre le trafic

lorsqu'il détecte des paquets ou des demandes suspectes vers un nombre quelconque de services différents exécutés sur un serveur.

Par défaut, Suricata fonctionne comme un système de détection d'intrusion (*Intrusion Detection System*) passif pour rechercher le trafic suspect sur un serveur ou un réseau.   
Il génére et enregistre des alertes pour une enquête plus approfondie.   
Il peut également être configuré comme un système de prévention des intrusions (Intrusion Prevention System) actif pour enregistrer, alerter et bloquer complètement le trafic réseau qui correspond à des règles spécifiques.

On peut déployer Suricata sur un hôte de passerelle dans un réseau afin d’analyser tout le trafic réseau entrant et sortant d'autres systèmes, ou on peut l'exécuter localement sur des hôtes individuels dans l'un ou l'autre mode.

## Conditions préalables

Selon la configuration de son réseau et la façon dont on prévoit utiliser Suricata, il est possible d’avoir besoin de plus ou moins de processeur (CPU) et de mémoire vive (RAM) pour le serveur.

En règle générale, plus il est prévu inspecter du trafic, plus on doit allouer de ressources à Suricata.

Dans un environnement de production, il est préférable de prévoir utiliser au moins 2 processeurs et 4 ou 8 Go de mémoire vive pour commencer.   
À partir de là, il est possible d’augmenter les ressources en fonction des performances de Suricata et de la quantité de trafic qu’il doit traiter.

Si on prévoit utiliser Suricata afin de protéger le serveur sur lequel il s'exécute, on aura besoin de …

* un serveur Debian 11 avec 2 processeurs ou plus ;
* un utilisateur sudo non root   
  et
* un pare-feu activé.

Sinon, si vous prévoyez d'utiliser Suricata sur un hôte de passerelle pour surveiller et protéger plusieurs serveurs, vous devrez vous assurer que la mise en réseau de l'hôte est correctement configurée.

# Installation de Suricata

Pour commencer à installer Suricata, il faut mettre à jour la liste des paquets disponibles sur le système Debian ...   
**>> sudo apt update**

Par la suite, on installe le paquet suricata …  
**>> sudo apt install suricata jq**

Une fois le paquet suricata installé, on active le service suricata afin qu'il s'exécute au redémarrage du système ...  
**>> sudo** **systemctl enable suricata.service**

Pour la suite des choses, il est préférable d’arrêter le service …  
**>> sudo** **systemctl stop suricata.service**

L'arrêt de Suricata garantit que lorsqu’on modifie et teste le fichier de configuration, toutes les modifications que l’on apporte seront validées et chargées au redémarrage de Suricata.

# Configuration de Suricata pour la première fois

Le paquet Suricata des référentiels OISF est livré avec un fichier de configuration qui couvre une grande variété de cas d'utilisation.

Le mode par défaut pour Suricata est le mode IDS. Aucun trafic ne sera abandonné mais seulement enregistré.   
Laisser ce mode par défaut est une bonne idée lors de la configuration initiale de Suricata.   
Une fois que l’on a configuré et intégré Suricata dans son environnement, et que l’on a une compréhension des types de trafic que Suricata doit alerter, il sera alors possible d'activer le mode IPS.

## Activation de l'ID de flux communautaire (Facultatif)

Suricata peut inclure un champ ID de communauté dans sa sortie JSON afin de faciliter la mise en correspondance d'enregistrements d'événements individuels avec des enregistrements dans des ensembles de données générés par d'autres outils.

Si on envisage d'éventuellement utiliser Suricata avec d'autres outils tels que Zeek ou Elasticsearch , il est préférable d’ajouter l'identifiant communautaire.

Pour activer cette option, on doit modifier le fichier de configuration /etc/suricata/suricata.yaml …  
**>> sudo** **nano /etc/suricata/suricata.yaml**

À la ligne 120 (# Community Flow ID), on doit modifier la valeur de la directive community-id pour true …  
**community-id: true**

**Remarque** …  
Avec l’utilitaire nano, on peut utiliser la fonction CTRL + - suivi du numéro de ligne afin de se déplacer rapidement à travers un fichier.

Maintenant, lors de l’examen des événements, ces événements possèderont un identifiant comme 1:S+3BA2UmrHK0Pk+u3XH78GAFTtQ=.

## Détermination de la ou des interfaces réseau à utiliser

On doit s’assurer que Suricata inspecte le trafic à l’aide de la bonne interface.

Par défaut, le fichier de configuration fourni avec le paquet OISF Suricata inspecte le trafic sur l’interface eth0.   
Si le système utilise une interface réseau par défaut différente ou si on souhaite inspecter le trafic sur plusieurs interfaces, il faudra modifier cette valeur.

Pour déterminer le nom de périphérique de l’interface réseau par défaut …  
**>> ip -p -j route show default  
[ {  
 "dst": "default",  
 "gateway": "192.168.1.1",  
 "dev": "ens33",  
 "flags": [ ]  
 } ]**

Le commutateur -p présente la sortie pour qu'elle soit plus lisible.  
Le commutateur -j affiche la sortie au format JSON.

La ligne dev indique le périphérique par défaut

On modifie le nom de l'interface dans la configuration de Suricata …  
**>> sudo** **nano /etc/suricata/suricata.yaml**

On recherche l’**élément af-packet:** (près de la ligne 580).   
On modifie la ligne pour qu'elle corresponde à son interface …  
**af-packet:  
 - interface: ens33  
(…)**

**Remarque** …  
Si on souhaite inspecter le trafic sur des interfaces supplémentaires, on peut ajouter d'autres objets …  
 - interface: <interface>

## Configuration du rechargement des règles en direct

Suricata prend en charge le rechargement des règles en direct, ce qui signifie qu’il est possible d’ajouter, supprimer et modifier des règles sans avoir à redémarrer le processus Suricata en cours d'exécution.

Pour activer l'option de rechargement en direct, on ajoute à la fin du fichier /etc/suricata/suricata.yaml les lignes suivantes …  
**>> sudo** **nano /etc/suricata/suricata.yaml  
detect-engine:  
 - rule-reload: true**

Avec ce paramètre en place, on pourra envoyer le signal système SIGUSR2 au processus en cours d'exécution.  
Suricata rechargera alors toutes les règles modifiées en mémoire.

Une commande suivante indiquera au processus Suricata de recharger ses ensembles de règles, sans toutefois redémarrer le processus …  
**>> sudo kill -usr2 $(pidof suricata)**

* La partie **sudo kill -usr2** de la commande utilise l'utilitaire kill afin d’envoyer le signal SIGUSR2 à l'ID de processus qui est renvoyé par le sous-shell.
* La partie **$(pidof suricata)** de la commande invoque un sous-shell et trouve l'ID de processus du démon Suricata en cours d'exécution.

On pourra utiliser cette commande à chaque exécution de la mise à jour de Suricata (suricata-update) ou de l’ajout de ou la modification de ses propres règles personnalisées.

# Mise à jour des ensembles de règles Suricata

À ce stade, si on devait démarrer Suricata, on devrait recevoir un message d'avertissement indiquant qu'il n'y a pas de règles chargées …  
**<Warning> - [ERRCODE: SC\_ERR\_NO\_RULES(42)] - No rule files match the pattern /etc/suricata/rules/suricata.rules**

Par défaut, le paquet Suricata inclut un ensemble limité de règles de détection (dans le répertoire /etc/suricata/rules), donc l'activation de Suricata à ce stade ne détecterait qu'une quantité limitée de mauvais trafic.

Suricata inclut un outil appelé suricata-update qui permet de récupérer des ensembles de règles auprès de fournisseurs externes.

Pour télécharger un ensemble de règles à jour pour Suricata …  
**>> sudo suricata-update --output /etc/suricata/rules**

Une image contenant texte

Description générée automatiquement   
**Cette sortie a été tronquée.**

Le commutateur --output (-o) de la commande précédente demande à l'outil de mise à jour d'enregistrer les règles dans un répertoire spécifique.

Les lignes en surbrillance indiquent qu'il a récupéré les règles ET Open (*Emerging Threat*s) gratuites et les a enregistrées dans le fichier de Suricata.  
La sortie indique également le nombre de règles qui ont été traitées et ajoutées.  
Certaines d’entre elles ont été activées.

# Ajout de fournisseurs d'ensembles de règles

L'outil suricata-update peut récupérer des règles auprès de divers fournisseurs d'ensembles de règles gratuits et commerciaux.  
Certains ensembles de règles comme l'ensemble ET Open sont disponibles gratuitement, tandis que d'autres nécessitent un abonnement payant.

Il est possible de répertorier l'ensemble de fournisseurs de règles par défaut en utilisant le commutateur list-sources de la commande suricata-update …  
**>> sudo suricata-update list-sources**

Une image contenant texte

Description générée automatiquement Vous recevrez une **Cette sortie a été tronquée.**

Par exemple, si on souhaite inclure l'ensemble de règles tgreen/hunting, on peut y arriver à l'aide de la commande suivante …  
**>> sudo suricata-update enable-source tgreen/hunting**

Une image contenant texte

Description générée automatiquement

On exécute à nouveau la commande suricata-update et le nouvel ensemble de règles sera ajouté, en plus des règles ET Open existantes et de toutes les autres qui ont été téléchargées …  
**>> sudo suricata-update --output /etc/suricata/rules**

# Validation de la configuration de Suricata

Après avoir modifié le fichier de configuration de Suricata afin d’inclure l'ID facultatif de communauté, spécifié l'interface réseau par défaut et activé le rechargement des règles en direct, il est judicieux d’effectuer un test de la configuration.

Suricata possède un mode de test intégré qui vérifiera la validité du fichier de configuration et de toutes les règles incluses.  
On peut ainsi valider ses modifications en utilisant le commutateur -T …  
**>> sudo** **suricata -T -c /etc/suricata/suricata.yaml -v**

Une image contenant texte

Description générée automatiquement

Le commutateur -v affiche des informations supplémentaires.  
Le commutateur -c indique à Suricata où trouver son fichier de configuration.

S'il y a une erreur dans le fichier de configuration, le mode test générera un code d'erreur et un message spécifiques que l’on peut utiliser afin d’aider à résoudre les problèmes.

**Remarque** …  
Le test peut prendre un certain temps (une minute ou deux) en fonction de la nombre et qualité de processeur allouée à Suricata et du nombre de règles que qui ont été ajoutées.

Voici un type d’erreur que l’on peut rencontrer …  
**18/04/2022 -- 15:10:21 - <Warning> - [ERRCODE: SC\_ERR\_NO\_RULES(42)] - No rule files match the pattern /etc/suricata/rules/test.rules**

Pour ce type d’erreur, on peut modifier le fichier de configuration pour inclure le chemin correct ou corriger les variables et les options de configuration non valides.

# Exécution de Suricata

Maintenant que l’on dispose d'une configuration et d'un ensemble de règles Suricata valides, on peut démarrer le serveur Suricata …  
**>> sudo** **systemctl start suricata.service icata**

On peut vérifier l'état du service …  
**>> sudo** **systemctl status suricata.service icata**

**Remarque** …  
Comme pour la commande du mode test, cela peut prendre une minute ou deux à Suricata pour charger et analyser toutes les règles.   
On peut utiliser la commande tail pour surveiller un message spécifique dans les journaux de Suricata indiquant qu'il a fini de démarrer …  
**>> sudo** **tail --follow /var/log/suricata/suricata.log**

Une image contenant texte

Description générée automatiquement

La ligne suivante indique que tout va bien …  
**18/4/2022 -- 10:03:04 - <Info> - All AFP capture threads are running.**

Cette ligne indique que Suricata est en cours d'exécution et prêt à inspecter le trafic.   
On peut quitter la commande tail avec CTRL+C.

# Test des règles Suricata

L'ensemble de règles ET Open qui ont été téléchargé contient plus de 30 000 règles.

On peut tester si Suricata détecte le trafic suspect avec la configuration qui a été générée.

On exécute la commande suivante pour générer une requête HTTP, qui renverra une réponse correspondant à la règle d'alerte de Suricata …  
**>> curl http://testmynids.org/uid/index.html**

La commande curl affiche une réponse semblable à ce qui suit …  
  
  
  
Cet exemple de données de réponse est conçu pour déclencher une alerte, en faisant semblant de renvoyer la sortie d'une commande comme id, commande qui pourrait s'exécuter sur un système distant compromis à l’aide un shell Web.

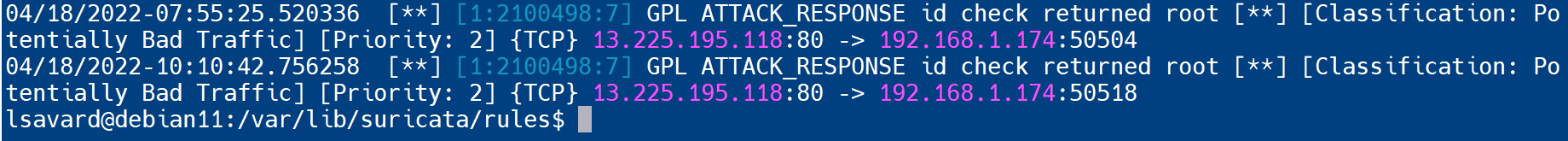
Il est maintenant possible de consulter les journaux de Suricata pour constater l’alerte correspondante.

Deux journaux sont activés avec la configuration par défaut de Suricata …

* /var/log/suricata/fast.log ;
* /var/log/suricata/eve.log (ouverture de session lisible par hôte).

## Examen du journal /var/log/suricata/fast.log

Pour rechercher une entrée de journal /var/log/suricata/fast.log correspondant à la demande curl, on doit utiliser la commande grep.   
À l'aide de l'identifiant de règle 2100498, on recherche les entrées qui lui correspondent …  
**>> grep 2100498 /var/log/suricata/fast.log**



Notez la **valeur 1:2100498:7** de la sortie, qui est l'ID de signature (sid) que Suricata utilise pour identifier une règle.

## Examen/var/log/suricata/eve.log

Suricata enregistre également les événements dans le journal /var/log/suricata/eve.log (surnommé le journal EVE) en utilisant JSON pour formater les entrées.

La documentation de Suricata recommande d'utiliser l'utilitaire jq pour lire et filtrer les entrées de ce fichier.   
Si ce n’est pas le cas, on doit installer jq …  
**>> sudo apt install jq**

Une fois jq installé, on peut filtrer les événements dans le journal EVE en recherchant la signature 2100498 …  
**>> jq 'select(.alert .signature\_id==2100498)' /var/log/suricata/eve.json**

Une image contenant texte

Description générée automatiquement  
**Cette sortie a été tronquée.**

La commande examine chaque entrée JSON et affiche toutes celles qui ont un objet alert, avec une clé signature\_id qui correspond à la valeur 2100498 que l’on recherche.

**Remarque** …  
La **signature\_id": 2100498** qui est la clé jq recherchée.   
La clé **community\_id": "1:orJE+IStTM2bjccd9RzqMmjYceE="** est l'identifiant de flux de communauté généré que l’on a activé dans le fichier de configuration de Suricata.

Chaque alerte générera un identifiant de flux communautaire unique.   
D'autres outils NMS peuvent également générer le même identifiant afin de permettre le recoupement d'une alerte Suricata avec la sortie d'autres outils.

Une entrée de journal correspondante dans l'un ou l'autre des fichiers journaux signifie que Suricata a inspecté avec succès le trafic réseau, l'a comparé à une règle de détection et a généré une alerte pour une analyse ou une journalisation ultérieure.

# Gestion des alertes Suricata

Une fois que l’on a configuré et testé les alertes, on peut choisir comment on souhaite les gérer.

Pour certains cas d'utilisation, la journalisation des alertes à des fins d'audit peut être suffisante.  
Pour d’autres cas, on peut possiblement préférerer adopter une approche plus active afin de bloquer le trafic des systèmes qui génèrent des alertes répétées.

Si on souhaite bloquer le trafic en fonction des alertes générées par Suricata, une approche consiste à utiliser les entrées du journal EVE, puis à ajouter des règles de pare-feu pour restreindre l'accès à un ou des systèmes.

On peut utiliser l'outil jq pour extraire des champs spécifiques d'une alerte, puis ajouter des règles UFW ou iptables pour bloquer les demandes.

# Structure des signatures Suricata

Les signatures Suricata peuvent sembler complexes au premier abord, mais une fois que l’on a appris comment elles sont structurées et comment Suricata les traite, on est en mesure de créer es propres règles en fonction des exigences de on réseau.

À un niveau élevé, les signatures Suricata se composent de trois parties …

* **Action** ;
* **En-tête** ;
* **Options**.

Une **action** qui spécifie ce que la règle effectue lorsque le trafic correspond à la règle.

Un **entête** qui décrit …

* les hôtes ;
* les adresses IP ;
* les ports ;
* les protocoles   
  et la direction du trafic (entrant ou sortant)

Les **options** qui spécifient des éléments tels que …

* l'ID de signature (sid) ;
* le message de journal ;
* les expressions régulières qui correspondent au contenu des paquets ;
* le type de classification   
  et
* d'autres modificateurs qui peuvent aider à affiner l'identification du trafic légitime et suspect.

La structure générale d'une signature est la suivante …  
**<Action> <Entête> <Options>  
<*Action*> <*Header*> <*Options*>**

Les parties d'en-tête et d'options d'une signature comportent plusieurs sections.   
Comme exemple … la règle avec le sid 2100498  
**alert ip any any -> any any (msg:"GPL ATTACK\_RESPONSE id check returned root"; content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:2100498; rev:7; metadata:created\_at 2010\_09\_23, updated\_at 2010\_09\_23;)**

La partie alert de la signature est l'action …  
alert ip any any -> any anysection est l' en- tête et le reste de la signature commençant par contient les options(msg:GPL ATTACK\_RESPONSE... de la règle .

Dans les sections suivantes, vous examinerez en détail chaque partie d'une règle Suricata.

Actions

La première partie de la sid:2100498signature est l'action, dans ce cas alert. La partie action d'une signature Suricata spécifie l'action à entreprendre lorsqu'un paquet correspond à la règle. Une action peut être l'une des suivantes selon que Suricata fonctionne en mode IDS ou IPS :

Pass - Suricata arrêtera d'analyser le paquet et l'autorisera, sans générer d'alerte.

Drop - Lorsque vous travaillez en mode IPS, Suricata arrête immédiatement de traiter le paquet et génère une alerte. Si la connexion qui a généré le paquet utilise TCP, il expirera.

Rejeter - Lorsque Suricata exécute le mode IPS, un paquet de réinitialisation TCP sera envoyé et Suricata supprimera le paquet correspondant.

Alerte - Suricata générera une alerte et l'enregistrera pour une analyse plus approfondie.

En-têtes

Chaque signature Suricata a une section d'en-tête qui décrit le protocole réseau, les adresses IP source et de destination, les ports et la direction du trafic. En se référant à l'exemple de sid:2100498signature, la section d'en-tête de la règle est la ip any any -> any anypartie en surbrillance :

sid:2100498

alert ip any any -> any any (msg:"GPL ATTACK\_RESPONSE id check returned root"; content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:2100498; rev:7; metadata:created\_at 2010\_09\_23, updated\_at 2010\_09\_23;)

Le format général de la section d'en-tête d'une règle est :

Format de règle

<PROTOCOL> <SOURCE IP> <SOURCE PORT> -> <DESTINATION IP> <DESTINATION PORT>

Le protocole peut être l'un des suivants :

TCP

UDP

ICMP

IP

Un certain nombre d'autres protocoles d'application

Les champs Source et Destination peuvent être des adresses IP ou des plages de réseaux, ou la valeur spéciale any, qui correspondra à toutes les adresses IP et tous les réseaux. La ->flèche indique le sens de la circulation.

Remarque : Les signatures peuvent également utiliser un marqueur non directionnel <>qui correspondra au trafic dans les deux sens. Cependant, la documentation de Suricata sur les marqueurs directionnels indique que la plupart des règles utiliseront la ->flèche de correspondance droite.

Si vous vouliez alerter sur le trafic sortant malveillant (c'est-à-dire le trafic quittant votre réseau), le champ Source serait alors l'adresse IP ou la plage réseau de votre système. La destination peut être l'adresse IP ou le réseau d'un système distant, ou la anyvaleur spéciale.

Inversement, si vous vouliez générer une alerte pour le trafic entrant malveillant, le champ Source pourrait être défini sur any, et la Destination sur l'adresse IP ou la plage réseau de votre système.

Vous pouvez également spécifier le port TCP ou UDP à examiner à l'aide des champs Port . Généralement, le trafic provenant d'un système se voit attribuer un port aléatoire, de sorte que la anyvaleur est appropriée pour le côté gauche de l' ->indicateur. Le port de destination peut également être anysi vous prévoyez d'examiner le contenu de chaque paquet entrant, ou vous pouvez limiter une signature pour analyser uniquement les paquets sur des ports individuels, comme 22 pour le trafic SSH ou 443 pour HTTPS.

L' ip any any -> any anyen-tête de sid:2100498est un en-tête générique qui correspondra à tout le trafic, quel que soit le protocole, les adresses IP source ou de destination ou les ports. Ce type d'en-tête fourre-tout est utile lorsque vous souhaitez vous assurer que le trafic entrant et sortant est contrôlé pour détecter tout contenu suspect.

Notez que les champs Source, Destination et Port peuvent également utiliser l' !opérateur de négation spécial, qui traitera le trafic qui ne correspond pas à la valeur du champ.

Par exemple, la signature suivante ferait de Suricata une alerte sur tous les paquets SSH entrants du anyréseau qui sont destinés à votre réseau (représenté par le 203.0.113.0/24bloc IP), qui ne sont pas destinés au port 22 :

Exemple d'en-tête

alert ssh any any -> 203.0.113.0/24 !22 (sid:1000000;)

Cette alerte ne serait pas très utile, car elle ne contient aucun message sur le paquet, ni un type de classification. Pour ajouter des informations supplémentaires à une alerte, ainsi qu'une correspondance sur des critères plus spécifiques, les règles Suricata ont une section Options dans laquelle vous pouvez spécifier un certain nombre de paramètres supplémentaires pour une signature.

Choix

Les arguments entre parenthèses (. . .) dans une signature Suricata contiennent diverses options et modificateurs de mots clés que vous pouvez utiliser pour faire correspondre des parties spécifiques d'un paquet, classer une règle ou consigner des messages personnalisés. Alors que les arguments d'en-tête d'une règle fonctionnent sur les en-têtes de paquet au niveau de l'adresse IP, du port et du protocole, les options correspondent aux données contenues dans un paquet.

Les options d'une règle Suricata doivent être séparées par un ;point-virgule et utilisent généralement un format clé:valeur. Certaines options n'ont pas de paramètres et seul le nom doit être spécifié dans une règle.

En utilisant l'exemple de signature de la section précédente, vous pouvez ajouter l' msgoption avec une valeur d' SSH traffic detected on non-SSH portexplication de l'objet de l'alerte :

Exemple d'en-tête

alert ssh any any -> 203.0.113.0/24 !22 (msg:"SSH TRAFFIC on non-SSH port"; sid:1000000;)

Une explication complète de la façon dont vous pouvez utiliser chaque option dans une règle Suricata dépasse le cadre de ce didacticiel. La documentation des règles Suricata commençant à la section 6.2 décrit chaque option de mot-clé en détail.

Cependant, il existe certaines options de base comme le contentmot-clé et divers mots-clés Meta qui sont utilisés dans la plupart des signatures, que nous examinerons dans les sections suivantes.

Le Contentmot clé

L'une des options les plus importantes pour toute règle est le mot- contentclé. Rappelez-vous l'exemple de sid:2100498signature :

sid:2100498

alert ip any any -> any any (msg:"GPL ATTACK\_RESPONSE id check returned root"; content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:2100498; rev:7; metadata:created\_at 2010\_09\_23, updated\_at 2010\_09\_23;)

La partie en surbrillance content:"uid=0|28|root|29|";contient le contentmot clé et la valeur que Suricata recherchera dans un paquet. Dans le cas de cet exemple de signature, tous les paquets provenant de n'importe quelle adresse IP sur n'importe quel port seront vérifiés pour s'assurer qu'ils ne contiennent pas la valeur de chaîne uid=0|28|root|29|(qui dans le tutoriel précédent a été utilisée comme exemple indiquant un hôte compromis).

Le contentmot-clé peut être utilisé avec la plupart des autres mots-clés de Suricata. Vous pouvez créer des signatures très spécifiques à l'aide de combinaisons d'en-têtes et d'options qui ciblent des protocoles d'application spécifiques, puis vérifier le contenu des paquets pour des octets individuels, des chaînes ou des correspondances à l'aide d'expressions régulières.

Par exemple, la signature suivante examine le trafic DNS à la recherche de tout paquet avec le contenu your\_domain.comet génère une alerte :

dns.query Exemple

alert dns any any -> any any (msg:"DNS LOOKUP for your\_domain.com"; dns.query; content:"your\_domain.com"; sid:1000001;)

Cependant, cette règle ne correspondrait pas si la requête DNS utilisait le domaine YOUR\_DOMAIN.COM, puisque Suricata utilise par défaut la correspondance de contenu sensible à la casse. Pour rendre les correspondances de contenu insensibles à la casse, ajoutez le nocase;mot clé à la règle :

DNS.query insensible à la casse Exemple

alert dns any any -> any any (msg:"DNS LOOKUP for your\_domain.com"; dns.query; content:"your\_domain.com"; nocase; sid:1000001;)

Désormais, toute combinaison de lettres minuscules ou majuscules correspondra toujours au mot- contentclé.

Le msgmot clé

Les exemples de signatures de ce didacticiel contiennent tous des msgmots clés contenant des informations sur une signature. Bien que l' msgoption ne soit pas obligatoire, si elle est vide, il est difficile de comprendre pourquoi une alerte ou une action de suppression s'est produite lors de l'examen des journaux de Suricata.

Une msgoption est conçue pour être une description textuelle lisible d'une alerte. Il doit être descriptif et ajouter du contexte à une alerte afin que vous ou quelqu'un d'autre qui analyse les journaux comprenne pourquoi l'alerte a été déclenchée. Dans la section [ referenceMot clé] ( referenceMot clé ) de ce didacticiel, vous découvrirez l' referenceoption que vous pouvez utiliser pour créer un lien vers plus d'informations sur une signature et le problème qu'elle est conçue pour détecter.

Les mots- clés sidetrev

Chaque signature Suricata nécessite un ID de signature unique ( sid). Si deux règles ont la même chose sid(dans l'exemple de sortie suivant, il s'agit de sid:10000000), Suricata ne démarrera pas et générera à la place une erreur comme celle-ci :

Example Duplicate sid Error

. . .

19/11/2021 -- 01:17:40 - <Error> - [ERRCODE: SC\_ERR\_DUPLICATE\_SIG(176)] - Duplicate signature "drop ssh any any -> 127.0.0.0/8 !22 (msg:"blocked invalid ssh"; sid:10000000;)"

. . .

Lorsque vous créez vos propres signatures, la plage 1000000-1999999 est réservée aux règles personnalisées. Les règles intégrées de Suricata sont comprises entre 2200000 et 2299999 . D'autres sidplages sont documentées sur la page Allocation SID des menaces émergentes .

L' sidoption est généralement la dernière partie d'une règle Suricata. Toutefois, s'il existe plusieurs versions d'une signature avec des modifications au fil du temps, une revoption est utilisée pour spécifier la version d'une règle. Par exemple, l'alerte SSH du début de ce didacticiel pourrait être modifiée pour analyser uniquement le trafic SSH sur le port 2022 :

Exemple de signature SSH avec rev

alert ssh any any -> 203.0.113.0/24 2022 (msg:"SSH TRAFFIC on non-SSH port"; sid:1000000; rev:2;)

La signature mise à jour inclut désormais l' rev:2option indiquant qu'elle a été mise à jour à partir d'une version précédente.

Le referencemot clé

Le referencemot-clé est utilisé dans les signatures pour décrire où trouver plus d'informations sur l'attaque ou le problème qu'une règle est censée détecter. Par exemple, si une signature est conçue pour détecter un nouveau type d'exploit ou de méthode d'attaque, le champ de référence peut être utilisé pour créer un lien vers un chercheur en sécurité ou le site Web d'une entreprise qui documente le problème.

La vulnérabilité Heartbleed dans OpenSSL est un exemple de bogue largement médiatisé et recherché. Suricata est livré avec une signature conçue pour vérifier les paquets TLS incorrects et inclut une référence à l' entrée principale Heartbleed CVE :

/etc/suricata/rules/tls-events.rules

alert tls any any -> any any (msg:"SURICATA TLS invalid heartbeat encountered, possible exploit attempt (heartbleed)"; flow:established; app-layer-event:tls.invalid\_heartbeat\_message; flowint:tls.anomaly.count,+,1; classtype:protocol-command-decode; reference:cve,2014-0160; sid:2230013; rev:1;)

Notez la partie en surbrillance reference:cve,2014-0160;de la signature. Cette option de référence vous indique, à vous ou à l'analyste qui examine les alertes de Suricata, où trouver plus d'informations sur le problème particulier.

L'option de référence peut utiliser n'importe lequel des préfixes du /etc/suricata/reference.configfichier. Par exemple, urlpourrait être utilisé à la place de cvedans l'exemple précédent, avec un lien direct vers le site Heartbleed à la place de l' 2014-0160identifiant CVE.

Le classtypemot clé

Suricata peut classer le trafic selon un ensemble préconfiguré de catégories qui sont incluses lorsque vous installez le package Suricata avec le gestionnaire de packages de votre distribution Linux. Le fichier de classification par défaut se trouve généralement dans /etc/suricata/classification.configet contient des entrées comme celles-ci :

/etc/suricata/classification.config

#

# config classification:shortname,short description,priority

#

config classification: not-suspicious,Not Suspicious Traffic,3

config classification: unknown,Unknown Traffic,3

config classification: bad-unknown,Potentially Bad Traffic, 2

. . .

Comme l'indique l'en-tête du fichier, chaque entrée de classification comporte trois champs :

Un nom court lisible par machine, dans les exemples ci-dessus not-suspicious, unknownet bad-unknownrespectivement.

La description d'une classification à utiliser avec les alertes, par exemple Not Suspicious Traffic.

Un champ prioritaire, qui détermine l'ordre dans lequel une signature sera traitée par Suricata. La priorité la plus élevée est la valeur 1. Les signatures qui utilisent un classificateur avec une priorité plus élevée seront vérifiées en premier lorsque Suricata traitera un paquet.

Dans l'exemple de sid:2100498signature, le type de classe est classtype:bad-unknown;, qui est mis en évidence dans l'exemple suivant :

sid:2100498

alert ip any any -> any any (msg:"GPL ATTACK\_RESPONSE id check returned root"; content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:2100498; rev:7; metadata:created\_at 2010\_09\_23, updated\_at 2010\_09\_23;)

La priorité implicite de la signature est 2, car il s'agit de la valeur attribuée au bad-unknowntype de classe dans /etc/suricata/classification.config. Si vous souhaitez remplacer la priorité par défaut d'un type de classe, vous pouvez ajouter une priority:noption à une signature, où nest une valeur comprise entre 1 et 255.

Le targetmot clé

Une autre option utile dans les signatures Suricata est l' targetoption. Il peut être défini sur l'une des deux valeurs suivantes : src\_ipet dest\_ip. Le but de cette option est d'identifier correctement les hôtes sourceet targetdans les journaux d'alertes de Suricata.

Par exemple, la signature SSH du début de ce didacticiel peut être améliorée avec l' target:dest\_ip;option :

Exemple de signature SSH avec champ cible

alert ssh any any -> 203.0.113.0/24 2022 (msg:"SSH TRAFFIC on non-SSH port"; target:dest\_ip; sid:1000000; rev:3;)

Cet exemple utilise dest\_ipparce que la règle est conçue pour vérifier le trafic SSH entrant dans notre exemple de réseau, c'est donc la destination. L'ajout de l' targetoiption à une règle entraînera les champs supplémentaires suivants dans la alertpartie d'une eve.jsonentrée de journal.

. . .

"source": {

"ip": "127.0.0.1",

"port": 35272

},

"target": {

"ip": "203.0.113.1",

"port": 2022

}

. . .

Avec ces entrées dans les journaux de Suricata, elles peuvent être envoyées à un outil de gestion des informations et des événements de sécurité (SIEM) pour faciliter la recherche d'alertes pouvant provenir d'un hôte commun ou d'attaques dirigées vers une cible spécifique sur votre réseau.

Conclusion

Dans ce didacticiel, vous avez examiné chacune des sections principales qui constituent une signature Suricata complète. Chacune des sections Actions , En- tête et Options d'une règle comporte plusieurs options et prend en charge l'analyse des paquets à l'aide de nombreux protocoles différents. Bien que ce didacticiel n'ait exploré aucune des sections en profondeur, la structure de la règle et les champs importants des exemples devraient suffire pour commencer à écrire vos propres règles.

Si vous souhaitez explorer des signatures complètes qui incluent beaucoup plus d'options que celles décrites dans ce didacticiel, explorez les fichiers du /etc/suricata/rulesrépertoire. S'il y a un champ dans une règle sur lequel vous aimeriez en savoir plus, la documentation des règles Suricata est la ressource faisant autorité sur la signification de chaque option et de ses valeurs possibles.

Une fois que vous êtes à l'aise pour lire et tester les signatures, vous pouvez passer au didacticiel suivant de cette série. Vous y apprendrez comment activer le mode IPS de Suricata, qui est utilisé pour supprimer le trafic suspect, par opposition au mode IDS par défaut qui ne génère que des alertes.

Référence …  
https://www.digitalocean.com/community/tutorials/how-to-install-suricata-on-debian-11