

# Suricata en tant que système de prévention des intrusions (IPS)

Par défaut, Suricata est configuré pour fonctionner comme un système de détection d'intrusion (IDS), qui ne génère que des alertes et enregistre le trafic suspect.   
Lors de l’activation du mode IPS, Suricata peut supprimer activement le trafic réseau suspect en plus de générer des alertes pour une analyse plus approfondie.

Avant d'activer le mode IPS, il est important de vérifier quelles signatures ont été activées et leurs actions par défaut. Une signature mal configurée ou encore une signature trop large peut entraîner la suppression du trafic légitime vers le réseau local, voire empêcher l’accès à ses serveurs à l‘aide d’une connexion SSH ou d'autres protocoles de gestion.

Les étapes de la mise en place de Suricata comme un Système de prévention d’Intrusions (IPS) sont ….

* Vérification des signatures installées et activées ;
* Ajout de ses propres signatures ;
* Sélection des signatures à utiliser en mode IPS ;
* Conversion des actions par défaut pour supprimer ou rejeter le trafic ;
* Envoi du trafic réseau par Suricata à l'aide de la cible netfilter NFQUEUE iptables, une fois les signatures en place ;
* Envoi et génération du trafic réseau non valide afin de s’assurer que Suricata le supprime comme prévu.

# Signatures personnalisées

Si on cherche à créer et inclure ses propres règles, il faut modifier le fichier /etc/suricata/suricata.yaml de Suricata pour inclure un chemin personnalisé vers ses signatures.

On identifie dans un premier temps les adresses IP publiques de son serveur afin de les utiliser dans les signatures personnalisées.

Pour afficher les adresses IP s’un serveur …  
**>> ip -brief address show  
  
**

On crée maintenant une signature personnalisée ayant comme objectif la recherche du trafic SSH vers les ports non SSH.  
on va, par la suite, l’inclure dans un nouveau fichier nommé /etc/suricata/rules/local.rules ...  
**>> sudo nano /etc/suricata/rules/local.rules  
alert ssh any any -> 192.168.1.174 !22 (msg:"SSH TRAFFIC on non-SSH port"; flow:to\_client, not\_established; classtype: misc-attack; target: dest\_ip; sid:1000000;)**

**alert ssh any any -> 2001:ABC::1/32 !22 (msg:"SSH TRAFFIC on non-SSH port"; flow:to\_client, not\_established; classtype: misc-attack; target: dest\_ip; sid:1000001;)**

**Remarque** …  
On remplace l'adresse IP publique de son serveur par ses adresses dans la règle.   
Si IPv6 n’Est pas utilisé, on peut ignorer l'ajout de la seconde signature dans cette règle et dans les règles suivantes.

Il est possible d’ajouter à cette étape des signatures personnalisées à ce fichier local.rules en fonction de son réseau et de ses applications.

Par exemple, si on souhaite provoquer une alerte pour le trafic HTTP vers des ports non standard …  
**alert http any any -> 192.168.1.174 !80 (msg:"HTTP REQUEST on non-HTTP port"; flow:to\_client, not\_established; classtype:misc-activity; sid:1000002;)**

**alert http any any -> 2001:ABC::1/32 !80 (msg:"HTTP REQUEST on non-HTTP port"; flow:to\_client, not\_established; classtype:misc-activity; sid:1000003;)**

Pour ajouter une signature qui vérifie le trafic TLS vers des ports autres TCP 443 pour un serveur Web …  
**alert tls any any -> 192.168.1.174 !443 (msg:"TLS TRAFFIC on non-TLS HTTP port"; flow:to\_client, not\_established; classtype:misc-activity; sid:1000004;)**

**alert tls any any -> 2001:ABC::1/32 !443 (msg:"TLS TRAFFIC on non-TLS HTTP port"; flow:to\_client, not\_established; classtype:misc-activity; sid:1000005;)**

Maintenant que l’on a défini des signatures personnalisées, il faut modifier le fichier de configuration /etc/suricata/suricata.yaml …  
**>> sudo nano /etc/suricata/suricata.yaml**

On recherche la section **rule-files:**. Elle devrait se retrouver près de la ligne 1861.

On modifie la section pour lui ajouter L,entrée local.rules : …  
**rule-files:  
 - suricata.rules  
 - local.rules**

On s’assure de valider la configuration de Suricata après avoir ajouté ses règles …  
**>> sudo suricata -T -c /etc/suricata/suricata.yaml -v**

Le test peut prendre un certain temps en fonction du nombre de règles que l’on a chargées dans le fichier /etc/suricata/rules/local.rules.   
Si le test semble trop long, on peut commenter la ligne **- suricata.rules** dans la configuration en ajoutant un # au début de la ligne et exécuter le test à nouveau.

**Remarque** …  
Il faut s’assurer de supprimer le commentaire (#) pour la ligne sudiicata.rules.

Une fois que les signatures sont validées, il faut exécuter l’outil suricata-update …  
**>> sudo suricata-update**

# Configuration des actions de signature

Maintenant que les signatures personnalisées ont été validées, on peut modifier l'action **alert** pour **drop** ou **reject**. Lorsque Suricata fonctionne en mode IPS, ces actions bloqueront activement le trafic invalide pour toute signature correspondante.

Le choix de l'action à utiliser est en fonction de ses besoins …

* Une **action drop** **supprimera immédiatement un paquet** et **tous les paquets suivants appartenant au flux** réseau ;
* Une **action reject enverra au client et au serveur un paquet de réinitialisation** si le trafic est basé sur TCP et un **paquet d'erreur ICMP** pour tout autre protocole.

On accède le ficher /etc/suricata/rules/local.rules pour modifier l'action alert au début de chaque ligne du fichier avec drop …  
**>> sudo nano /etc/suricata/rules/local.rules  
drop ssh any any -> 192.168.1.174 !22 (msg:"SSH TRAFFIC on non-SSH port"; classtype: misc-attack; target: dest\_ip; sid:1000000;)  
(…)**

**Remarque** …  
Si on a exécuté la commande suricata-update précédemment, on pourrait avoir plus de 30 000 signatures incluses dans le fichier suricata.rules.

Si on convertit, à l’aide d’un script, chaque signature pour **drop** ou **reject**, on risque de bloquer l'accès légitime à son réseau ou à ses serveurs.   
Au lieu de cela, il est préférable de laisser les règles du fichier suricata.rules pour le moment et on ajoute ses signatures personnalisées à local.rules.   
Suricata continuera à générer des alertes pour le trafic suspect décrit par les signatures du fichier suricata.rules pendant son exécution en mode IPS.

Après avoir collecté quelques jours ou semaines d'alertes, on pourra les analyser et choisir les signatures pertinentes à convertir pour drop ou reject en fonction de leur sid.

Une fois que toutes les signatures ont été configurée avec l'action souhaitée, l'étape suivante consiste à reconfigurer puis à redémarrer Suricata en mode IPS.

# Mode d'activation nfqueue

Suricata fonctionne en mode IDS par défaut, ce qui signifie qu'il ne bloquera pas activement le trafic réseau.

Afin de basculer en mode IPS, on doit modifier le fichier /etc/default/suricata de configuration de Suricata.  
**>> sudo nano /etc/default/suricata**

On recherche la ligne LISTENMODE=af-packet et on la commante à l’aide d’un # au début de la ligne.   
On ajoute ensuite une nouvelle ligne LISTENMODE=nfqueue qui indique à Suricata de s'exécuter en mode IPS …  
**# LISTENMODE=af-packet  
LISTENMODE=nfqueue**

On doit redémarrer Suricata …  
**>> sudo systemctl restart suricata.service**

On vérifie le statut de Suricata …  
**>> sudo systemctl status suricata.service**

# Configuration d'UFW pour envoyer du trafic vers Suricata

Maintenant que Suricata a été configuré pour traiter le trafic en mode IPS, l'étape suivante consiste à diriger les paquets entrants vers Suricata.

On s’assure que l’utilitaire UFS (Uncomplicated Firewall) est installé …   
**>> sudo apt install ufw**

Pour ajouter les règles requises pour Suricata à UFW, on doit modifier les fichiers du pare-feu …

* /etc/ufw/before.rules (règles IPv4)   
  et
* /etc/ufw/before6.rules(IPv6) directement.

On accède au fichier de configuration d’UFW …  
**>> sudo nano /etc/ufw/before.rules**

Vers le début du fichier, on insère les lignes suivantes …  
**(…)  
# Don't delete these required lines, otherwise there will be errors  
\*filter  
:ufw-before-input - [0:0]  
:ufw-before-output - [0:0]  
:ufw-before-forward - [0:0]  
:ufw-not-local - [0:0]  
# End required lines**

**## Start Suricata NFQUEUE rules  
-I INPUT 1 -p tcp --dport 22 -j NFQUEUE --queue-bypass  
-I OUTPUT 1 -p tcp --sport 22 -j NFQUEUE --queue-bypass  
-I FORWARD -j NFQUEUE  
-I INPUT 2 -j NFQUEUE  
-I OUTPUT 2 -j NFQUEUE  
## End Suricata NFQUEUE rules**

**Remarque** …  
Si on utilise le protocole IPv6, on doit également ajouter les mêmes lignes en surbrillance à la même section du fichier /etc/ufw/before6.rules.  
On doit s’assurer que les deux fichiers ont le même contenu.

Les deux premières règles INPUT et OUTPUT sont utilisées pour contourner Suricata afin que l’on puisse se connecter au serveur en utilisant SSH, même lorsque Suricata n'est pas en cours d'exécution.   
Sans ces règles, une signature incorrecte ou trop large pourrait bloquer les accès SSH.   
De plus, si Suricata est arrêté, tout le trafic sera envoyé à la cible NFQUEUE, puis abandonné puisque Suricata n'est pas en cours d'exécution.

La règle FORWARD garantit que si le serveur agit comme passerelle pour d'autres systèmes, tout ce trafic passera également par Suricata pour traitement.

Les deux dernières règles INPUT et OUTPUT envoient tout le trafic restant qui n'est pas du trafic SSH à Suricata pour traitement.

On doit redémarrer UFW pour charger les nouvelles règles …  
**>> sudo systemctl restart ufw.service**

**Remarque** …  
Si on utilise un autre pare-feu, il faudra modifier ces règles pour qu'elles correspondent au format attendu par le pare‑feu.

Si on utilise iptables, on peut insérer ces règles directement à l'aide des commandes iptables et ip6tables.   
Cependant, vous on devra s’assurer que les règles sont persistantes lors des redémarrages avec un outil tel que iptables‑persistent.

Si on utilise firewalld, les règles suivantes dirigeront le trafic vers Suricata …  
**firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 0 -p tcp --dport 22 -j NFQUEUE --queue-bypass  
firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 1 -j NFQUEUE  
firewall-cmd --permanent --direct --add-rule ipv6 filter INPUT 0 -p tcp --dport 22 -j NFQUEUE --queue-bypass  
firewall-cmd --permanent --direct --add-rule ipv6 filter INPUT 1 -j NFQUEUE  
  
firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -j NFQUEUE  
firewall-cmd --permanent --direct --add-rule ipv6 filter FORWARD 0 -j NFQUEUE  
  
firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 0 -p tcp --sport 22 -j NFQUEUE --queue-bypass  
firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 1 -j NFQUEUE  
firewall-cmd --permanent --direct --add-rule ipv6 filter OUTPUT 0 -p tcp --sport 22 -j NFQUEUE --queue-bypass  
firewall-cmd --permanent --direct --add-rule ipv6 filter OUTPUT 1 -j NFQUEUE**

À ce stade du didacticiel, Suricata est configuré pour s'exécuter en mode IPS et le trafic réseau est envoyé à Suricata par défaut.

On peut redémarrer le serveur à tout moment et les règles Suricata et du pare-feu seront persistantes.

# Test du trafic invalide

On peut valider si Suricata supprime les paquets correspondant aux signatures personnalisées et autres incluses.

Comme exemple …  
**drop 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;)**

On recherche et on modifie la règle dans le fichier /etc/suricata/rules/suricata.rules pour utiliser l'action drop si la signature y est incluse.   
Sinon, on ajoute la règle au fichier.

On envoie le signal SIGUSR2 à Suricata pour qu'il recharge ses signatures …  
**>> sudo kill -usr2 $(pidof suricata)**

On peut maintenant tester la règle en utilisant l’utilitaire curl …  
**>> curl --max-time 5 http://testmynids.org/uid/index.html**

On devrait recevoir une erreur indiquant que la requête a expiré, ce qui indique que Suricata a bloqué la réponse http …  
**curl: (28) Operation timed out after 5000 milliseconds with 0 out of 39 bytes received**

On peut confirmer que Suricata a supprimé la réponse HTTP en utilisant l’utilitaire jq pour examiner le fichier eve.log …  
**>> jq 'select(.alert .signature\_id==2100498)' /var/log/suricata/eve.json**

On devrait recevoir une sortie comme suit …  
**{  
. . .  
 "community\_id": "1:tw19kjR2LeWacglA094gRfEEuDU=",  
 "alert": {  
 "action": "blocked",  
 "gid": 1,  
 "signature\_id": 2100498,  
 "rev": 7,  
 "signature": "GPL ATTACK\_RESPONSE id check returned root",  
 "category": "Potentially Bad Traffic",  
 "severity": 2,  
 "metadata": {  
 "created\_at": [  
 "2010\_09\_23"  
 ],  
 "updated\_at": [  
 "2010\_09\_23"  
 ]  
 }  
 },  
 "http": {  
 "hostname": "testmynids.org",  
 "url": "/uid/index.html",  
 "http\_user\_agent": "curl/7.68.0",  
 "http\_content\_type": "text/html",  
 "http\_method": "GET",  
 "protocol": "HTTP/1.1",  
 "status": 200,  
 "length": 39  
 },  
. . .**

La ligne en surbrillance action: "blocked" confirme que la signature correspond et Suricata a abandonné ou rejeté la requête HTTP de test.

**Référence** …  
https://www.digitalocean.com/community/tutorials/how-to-configure-suricata-as-an-intrusion-prevention-system-ips-on-ubuntu-20-04