Snort Inline — Deuxième partie

Gazette Linux n°118 — Septembre 2005

Pete Savage

Article paru dans le n°118 de la Gazette Linux de septembre 2005.
Traduction française par Deny <deny CHEZ monaco POINT net>.
Relecture de la traduction française par Joëlle Cornavin <jcornavi CHEZ club TIRET internet POINT fr>.

Le mois dernier, nous avons réussi à compiler et installer avec succès snort_inline, l'adaptation IPS (*Intrusion Prevention System*, système de prévention d'intrusion) du paquetage commun snort. Ce mois-ci, cependant, nous porterons attention à quelques-unes des tâches de maintenance qui permettent de garder notre système de prévention d'intrusion opérationnel et à jour.

Cette seconde partie se concentrera sur les points suivants :

- La mise à jour automatique des règles snort.
- La conception d'un *script* pour pouvoir redémarrer snort aisément et le démarrer au moment de l'amorçage.

Dans l'article du mois dernier, nous avons converti toutes nos règles snort d'acceptation en règles de rejet. De temps en temps, il est utile de lire attentivement les journaux, car il y a toujours de faux positifs transmis dans la communauté snort. Si vous découvrez qu'une règle amène un trafic normal à être rejeté, vous pouvez toujours changer le type d'action des règles de rejet en alerte.

Même si vous avez convenablement actualisé votre application snort_inline à la fin du mois dernier, beaucoup de choses se sont passées depuis lors. Les auteurs de virus sont loin d'être restés inactifs et les programmeurs *shell* ont codé fiévreusement. Le monde étant ce qu'il est, il est important que vous gardiez à jour les règles de votre application snort. Ce processus peut être effectué manuellement ou avec l'aide d'un petit paquetage appelé OinkMaster.

Le site *web* snort.org (http://www.snort.org) offre trois types différents de téléchargement pour la mise à jour des règles : un abonnement (payant), être enregistré ou non enregistré. Comme les règles si l'on est pas enregistré ne sont mises à jour qu'au moment de la toute dernière version majeure sur snort.org (http://www.snort.org) et que les règles via abonnement sont payantes, nous avons choisi d'utiliser les règles en tant qu'utilisateur enregistré pour cet article.

Avant de pouvoir continuer, vous devez créer un compte sur snort.org (http://www.snort.org). Au bas de la page de préférences utilisateur, cliquez sur le bouton d'ajout de code pour générer votre code snort unique. Nous allons employer un code fictif pour les besoins de cet article, *587cba039g9f03300f9e98b*.

Votre code snort unique vous permet de télécharger les dernières règles snort en tant qu'utilisateur enregistré. Il faudra modifier le fichier de configuration de OinkMaster, pour qu'il télécharge ces toutes dernières règles pour nous. Veuillez suivre les étapes ci-dessous pour obtenir OinkMaster, puis commencer la configuration. À titre de référence, nous avons utilisé le fichier `oinkmaster-1.2.tar.gz`. Comme dans l'article précédent, nous partirons du principe que tous les fichiers seront téléchargés dans `/home/snort/`.

Grâce à la commande ci-dessus, nous avons déplacé le *script* oinkmaster dans le répertoire `/usr/local/bin`. Ce dernier peut aller à l'endroit de votre choix, mais pour la suite de cet article, nous supposerons qu'il réside dans `/usr/local/bin/`. Les mêmes règles s'appliquent au fichier `oinkmaster.conf`, sauf qu'il a été déplacé dans `/etc/`. Nous pouvons à présent modifier le fichier de configuration de OinkMaster, pour tirer profit de notre enregistrement chez snort. Ouvrez le fichier `/etc/oinkmaster.conf` dans votre éditeur de texte favori. Vers la ligne 61, vous devriez modifier la ligne :

... et remplacer `<oinkcode>` par la clé que vous venez d'acquérir ci-dessus. Nous emploierons notre clé factice mentionnée précédemment, à titre d'exemple.

OinkMaster devrait à présent être en mesure de télécharger toutes les mises à jour de snort disponibles. Nous effectuerons une autre modification, afin que OinkMaster change toutes les règles d'alerte en règles de rejet. Les implications en ont déjà été évoquées précédemment. Il peut en fait y avoir quelques règles indésirables de type « rejet ».

Vers la ligne 292 dans le fichier `oinkmaster.conf`, vous devriez lire le commentaire suivant :

Ces lignes se rapportent directement à notre implémentation de snort et nous montrent un exemple de la manière de modifier une règle d'alerte en rejet. Il est intéressant de noter, à ce stade, que toutes les règles officielles de snort sont dotées d'un SID (*Signature ID*, identifiant de signature). C'est une valeur unique utilisée pour identifier une règle parmi d'autres et autoriser des notes de version pour chaque règle. De cette façon, nous pouvons apporter une modification à une règle particulière. Nous ajouterons maintenant une ligne au-dessous de celle-ci, pour convertir toutes nos règles d'alerte en règles de rejet.

L'astérisque sur la ligne ci-dessus demande à OinkMaster d'appliquer la modification à toutes les règles qu'il télécharge.

Si nous créons maintenant un répertoire temporaire, nous pouvons tester notre configuration OinkMaster et contrôler si nos règles ont été téléchargées et modifiées.

Après avoir lancé les commandes ci-dessus, vous devriez obtenir une sortie similaire à celle affichée ci-après. Cette sortie a été abrégée afin d'économiser de l'espace.

D'après la sortie ci-dessus, nous pouvons affirmer que OinkMaster a modifié l'ensemble des 3 190 règles et ajouté toutes les règles qui n'étaient pas déjà présentes : dans notre cas, il s'agissait de la totalité de celles-ci. À présent, exécutez la commande suivante ;

Après quelques commentaires au sujet de la licence, vous devriez voir deux règles, comme ci-dessous. Vous remarquez que le premier mot de chaque ligne est `drop`, ce qui témoigne du succès de notre modification du SID.

Nous sommes en mesure de télécharger ces règles dans notre propre répertoire de règles snort_inline du mois dernier. Je vous conseille fortement de faire une sauvegarde de votre répertoire des règles avant de continuer plus avant. Je vous suggère également de déplacer toutes les règles personnalisées, avec lesquelles vous avez pu faire des expérimentations, dans un fichier appelé `local.rules`. C'est parce que OinkMaster supprimera toute règle qui n'est pas dans la version officielle, à moins qu'elle ne soit présente dans un fichier séparé. Celle-ci constituera une modification à apporter au fichier `snort_inline.conf`, ce que nous ferons dans un instant. La commande ci-dessous effectuera le même processus que précédemment, mais elle modifiera dorénavant nos règles snort_inline existantes.

La sortie devrait être similaire sinon identique à celle de tout à l'heure. Si à présent vous exécutez snort_inline, comme nous l'avons fait précédemment, à l'aide de la commande :

... il se peut que vous voyiez s'afficher l'erreur suivante :

C'est parce que le nouvel ensemble de règles utilise une règle où le nombre de mots-clés *flowbit* excède celui de la configuration de snort_inline. Les *flowbits* servent à conserver la trace de l'état d'une connexion TCP/IP. Nous pouvons modifier une ligne dans la configuration de snort_inline pour résoudre ce problème.

Ouvrez `/etc/snort_inline/snort_inline.conf` dans votre éditeur favori et ajoutez la ligne suivante :

Si vous tentez maintenant de lancer à nouveau snort_inline, vous devriez voir s'afficher le message familier ;

Si tel est le cas, vous venez d'employer OinkMaster pour mettre à jour votre ensemble de règles snort_inline avec les plus récentes disponibles sur snort.org (http://www.snort.org).

Il serait avantageux, cependant, de faire faire ces mises à jour automatiquement, de façon à ce que le processus nous soit transparent. Nous allons ajouter ce processus à la routine crontab et nous assurer qu'elle se produit chaque nuit. Pour ce faire, nous lancerons la commande suivante :

Cette commande ouvrira l'ordonnanceur de tâches cron nocturnes de l'utilisateur actuel, qui doit être *root*. Nous allons faire appel à une version légèrement modifiée de la ligne de commande utilisée précédemment pour démarrer OinkMaster. Habituellement, lors du lancement d'un processus en tant que tâche cron, toute sortie provenant de ce processus fait l'objet d'un message électronique envoyé au propriétaire de la tâche cron. Bien que cela puisse être utile, il peut aussi être ennuyeux de recevoir chaque nuit un message électronique avec le même contenu. Nous allons supprimer la sortie provenant du processus de OinkMaster, afin d'économiser de l'espace dans notre boîte aux lettres. Vous devrez ajouter une ligne similaire à celle qui suit dans votre fichier crontab :

Cette ligne exécutera la mise à jour à 00:55 chaque nuit. Comme la majorité de la population dort à cette heure, c'est un bon moment pour effectuer les tâches de maintenance. L'expression `> /dev/null 2>&1` ajoutée à la fin de cette ligne route toute la sortie vers le périphérique `/dev/null`, en l'ignorant efficacement. Après avoir enregistré et quitté l'application, vous devriez voir une ligne similaire à la ligne suivante :

Elle indique que votre nouvelle tâche cron a été installée avec succès.

Il serait temps d'essayer de créer le fichier `local.rules` et de le lier à notre fichier de configuration snort_inline. Suivez les étapes ci-dessous pour créer le fichier `local.rules` :

Ouvrez ce fichier dans votre éditeur favori et ajoutez-y la ligne suivante ;

Cela fonctionnera pratiquement de la même façon que lors du test de la première partie : il serait utile de revenir à cet article pour vous aider ici. Nous devons redémarrer le processus snort comme

précédemment.

Bien qu'on s'écarte légèrement du sujet, il est bon de noter que certaines parties de snort_inline ne sont pas, encore, tout à fait complètes. Je me réfère à la gestion par snort_inline de certaines connexions *shell*. Au moment de l'écriture de cet article, il y a un problème avec, par exemple, la concordance des règles telnet. Le problème se produit en raison de l'ordre dans lequel les paquets sont traités. Dans la version originale de snort, la carte réseau est placée en mode espion (*promiscuous mode*), ce qui permet de voir tout le trafic réseau, mais non de le modifier.

Ce comportement est analogue à l'image d'un homme muni d'une tablette électronique derrière un tapis roulant présentant des articles. L'homme s'assied et note tout article qu'il croit défectueux ou à problème et en fait part à son supérieur. Une fois que les pièces atteignent l'extrémité du tapis roulant, elles sont assemblées pour en faire un article complet.

En revanche, snort_inline interfère réellement dans le processus du tapis roulant. snort_inline sélectionne les éléments et les examine lui-même. C'est à ce stade qu'il détermine si leur contenu concorde. La version originale de snort possède aussi un mécanisme appelé stream4_reassemble. Ce dernier est analogue à l'image d'un second homme inspectant les pièces au fur et à mesure qu'elles sont assemblées, à l'extrémité du tapis roulant. Ce second contrôleur de qualité peut alors examiner l'assemblage entier pour voir s'il y a un problème.

Le problème est que snort_inline fonctionne avec seulement un homme au centre du tapis roulant. Par conséquent des connexions comme telnet, où chaque caractère saisi est envoyé en tant que paquet séparé, peuvent contourner certaines règles.

Ceci se produit parce que le rejet d'une connexion quand, par exemple un utilisateur saisit **`to su root`** (une règle en fait de snort, SID 715), exigerait le réassemblage du flux de données avant que le contenu correspondant puisse être traité. L"homme au centre verra juste `t -- o -- -- s -- u -- -- r -- o -- o -- t`, toute cette saisie paraissant parfaitement acceptable. Le snort originel verrait le flux **`to su root`** assemblé à l'extrémité du tapis roulant, ce qui aurait alerté le surveillant.

C'est la raison pour laquelle certaines des règles de snort_inline ne fonctionnent pas réellement, même à un niveau `alert`. D'aucuns ont suggéré de lancer deux versions de snort, une pour le rejet et l'autre pour l'alerte. La manière de choisir d'implémenter et de résoudre ce problème est de votre ressort. Cependant, des rumeurs courent selon lesquelles les développeurs de snort_inline travailleraient sur cet aspect, ce qui exigerait la manipulation de paquets déclassés (*out-of-sequence*).

Ne serait-il pas pratique de pouvoir démarrer, arrêter et redémarrer snort en cliquant sur un bouton ? Peut-être ne pouvons-nous pas le faire avec juste un bouton, mais la méthode actuelle prend trop de temps. Il serait agréable d'avoir un *script* de démarrage qui veillerait au chargement de ip_queue, à la création des règles iptables et au lancement du démon snort_inline. J'ai écrit un *script* très simple pour exécuter cette procédure, avec un système Red Hat/Fedora à l'esprit. Dans ce cas, il ne fonctionnera pas sur la plupart des autres distributions. Ce n'est pas difficile de créer un *script* init.d. Prenez en référence un programme courant et modifiez-le comme nécessaire pour en apprendre les bases.

Jetons un bref coup d'œil au script ci-dessous, pour déterminer ce que fait chaque partie et comment elle s'associe avec notre processus usuel, pour démarrer et arrêter snort_inline. Les premières lignes, précédées d'un caractère #, sont des commentaires. Elles seront ignorées lors de l'exécution du script, mais certaines ont nécessaires afin d'initialiser la séquence de démarrage de snort_inline. Il y a deux commandes sources qui chargent les bibliothèques standard. La première ligne qui nous intéresse est la suivante :

Cette ligne vérifie si le fichier binaire snort_inline existe réellement. Dans le cas contraire, le *script* s'interrompt.

La fonction `start` vérifie d'abord si le module ip_queue est chargé. S'il ne l'est pas, il fait alors appel à modprobe pour le charger. Comme vous le constatez, la plupart des commandes du *script* de démarrage sont semblables, sinon identiques, à celles employées dans l'article du mois dernier. L'instruction **echo** dans ce script ne sert ici qu'à produire les lignes `[ OK ]/[ FAILED ]` affichées lors du lancement d'un script init.d.

Les quelques commandes suivantes diffèrent de notre procédure standard. Au lieu de simplement créer une seule règle pour iptables, comme nous l'avons fait précédemment, nous allons créer un groupe iptables. Ce groupe servira de collection pour toutes nos règles iptables pour snort_inline. Pour quelles raisons me direz-vous ? Quand il arrive que le service snort_inline s'arrête, nous voulons que nos règles iptables soient supprimées également. À défaut de vider l'ensemble de règles iptables entier, nous pouvons simplement vider notre nouveau groupe pour supprimer nos règles snort_inline.

Ces trois lignes effectuent les actions suivantes ;

Ligne 1 : crée un nouveau groupe iptables appelé ip_queue
Ligne 2 : route tout le flux entrant tcp vers ce nouveau groupe
Ligne 3 : ajoute notre règle par défaut pour le port 80 à envoyer à la ip_queue

Pour finir, le binaire snort_inline est chargé à l'aide d'une ligne de commande tout à fait classique. Notez que le mot démon a été ajouté en début de ligne, ce qui démarre snort_inline en mode tâche de fond, séparant l'application du terminal.

La fonction `stop` effectue toutes ces tâches, mais à l'inverse. Autrement dit, d'abord le processus snort_inline est arrêté, puis les règles iptables sont vidées avant la suppression finale du module ip_queue. Notez la manière dont les règles iptables sont vidées.

Ligne 1 : vide (supprime) toutes les règles du groupe ip_queue
Ligne 2 : supprime le lien depuis `INPUT queue` vers le groupe ip_queue
Ligne 3 : supprime le groupe ip_queue

Comme on peut le voir dans le *script* ci-dessous, il existe aussi une fonction `restart`, qui lance la fonction `stop`, immédiatement suivie de la fonction `start`. Ce comportement nous facilitera considérablement la vie à partir de maintenant. Nous sommes dorénavant en mesure d'ajouter snort_inline à la séquence de démarrage, ce qui permet de l'exécuter au moment de l'amorçage. Vous devriez à présent copier le *script* depuis l'emplacement ci-dessous ou depuis ce lien (outils/lg118-F/snort_inline.txt), puis le placer dans le répertoire `/etc/init.d/` et l'appeler snort_inline. Il faudra également modifier les droits d'accès en `755` avec chmod, le fichier devant appartenir à la fois au groupe et au super-utilisateur (*root*), sinon le service ne l'utilisera pas. Si vous avez des doutes sur la manière de procéder, une fois le fichier créé dans `/etc/init.d/`, exécutez les commandes suivantes :

Vous devriez maintenant tester le *script* de démarrage, pour vous assurer qu'il fonctionne réellement. Vérifier d'abord que snort_inline ne tourne pas, que les règles iptables ont été supprimées et que ipqueue a été arrêté. Exécutez ensuite la commande suivante :

Votre systéme snort_inline devrait alors afficher les lignes suivantes :

Du moment que vous voyez apparaître `OK` sous la ligne, vous devriez avoir réussi. S'il subsiste quelques avertissements `[ FAILED ]`, vous devriez réexécuter chaque commande individuellement. Si vous avez

saisi le *script* à nouveau vous-même, cherchez les erreurs. Sinon, vérifiez si vous avez apporté des changements quelconques à des répertoires ou à la configuration du mois dernier.

Nous allons à présent permettre l'exécution de snort_inline au moment de l'amorçage. N'oubliez pas d'ajouter les lignes de commentaire et de description de chkconfig, comme exigé par chkconfig pour pouvoir l'ajouter à la séquence de démarrage. La saisie de la commande ci-dessous ajoutera snort_inline au gestionnaire de services dans Red Hat/Fedora :

Cette ligne ajoute snort_inline dans la base de données des services. La saisie ensuite de :

demandera au démon init de charger snort_inline, aux niveaux d'exécution 234 et 5. Pour pouvoir le tester, vous allez devoir redémarrer le serveur. Vous devriez, pendant la séquence de démarrage, voir la sortie du vidage précédent, indiquant que les services ont été démarrés correctement. Par ailleurs, la ligne :

détermine les niveaux d'exécution initiaux dans lesquels snort_inline devrait être démarré, suivi des priorités minimales de démarrage et d'arrêt. Ces valeurs conviennent très bien sur ce serveur de test, elles ne devraient pas vous causer de problème.

Veuillez notez que c'est un *script* très simple, à ne pas considérer comme un *script* init.d définitif. Il s'occupe simplement des bases et permet un redémarrage aisé du processus snort_inline.

Pour finir, nous demanderons à crontab de redémarrer le démon snort_inline, une fois que le téléchargement des règles via OinkMaster a été achevé. Le *script* de OinkMaster a été programmé pour se lancer à 00:55. Nous programmerons snort_inline pour qu'il redémarre une demi-heure plus tard. Exécutez la commande suivante :

Modifiez à présent ce document, en plaçant la ligne de redémarrage de snort_inline sous la commande de OinkMaster :

Les lignes ci-dessus redirigent toute leur sortie vers le processus `null`, l'ignorant donc pour l'essentiel. Il se peut que vous soyez amené à voir la sortie de ces processus, auquel cas supprimez le `< /dev/null 2>&1` des lignes ci-dessus. Vous pouvez parfaitement modifier la périodicité. Cependant, à fins de tests, c'est probablement la meilleure chose à faire.

Voici le *script* complet :

Donc, en résumé, je vous dois des excuses pour ne pas avoir abordé les règles personnalisées. C'est sur mon agenda du mois prochain pour la troisième partie de snort_inline. J'ai reçu plusieurs messages électroniques à propos de deux règles personnalisées et, effectivement, des moyens simples pour démarrer et redémarrer snort_inline.

Pete programme depuis l'âge de 10 ans sur un vieil Atari 800 XE©. Bien qu'il ait obtenu le diplôme d'ingénieur acoustique à la ISVR mondialement connue, à Southampton (Royaume-Uni), l'appel de la programmation lui a fait rebrousser chemin et depuis, il travaille comme développeur *web*. Il utilise à la fois les plates-formes Linux et Windows. Il vit toujours au Royaume-Uni, où il coule actuellement des jours heureux avec son épouse.