TP 2 - Configuration de Snort et mise en œuvre de signatures

0. Initialisation du TP

Tout d'abord, **sauvegardez le fichier etc/snort.conf dans snort.conf.bak**. Ensuite, afin d'améliorer les performances de Snort et d'éviter la levée de trop d'alertes, supprimez toutes les lignes se trouvant entre `include $RULE_PATH/bad-traffic.rules` jusqu'à `include $RULE_PATH/experimental.rules` du fichier `etc/snort.conf`. Il ne doit rester qu'une inclusion pour `local.rules`.

1. Utilisation de SNORT comme un IDS

Dans le mode IDS, SNORT n'enregistre pas tous les paquets capturés comme dans le mode sniffeur mais applique un ensemble de règles sur chacun d'eux. Si un paquet coïncide avec une règle, le paquet est alors enregistrer ou une alerte est déclenchée. Sinon, le paquet est supprimé. Pour cela, il faut spécifier au lancement de SNORT un fichier de configuration des différents composants de SNORT (préprocesseurs, règles du moteur de détection, …).

```
snort -dev -l monrepdelog -c /etc/snort/snort.conf –i <interface>
```

Avec snort.conf, le nom du fichier de configuration de Snort. Ceci appliquera l'ensemble des règles sélectionnées aux paquets analysés pour décider si une action doit être prise ou pas. Si un répertoire de sortie n'est pas spécifié, il s'agit par défaut de /etc/log/snort. Si la sonde Snort, fonctionne sur un long terme il n'est pas nécessaire d'avoir un affichage à l'écran qui consomme du temps et de la puissance. Il n'est également pas nécessaire d'enregistrer les entêtes pour la plupart des applications. Donc sans les options e et v. En utilisant l'option –A il est possible de rediriger les alertes à l'écran.

```
snort –A console -l monrepdelog -c /etc/snort/snort.conf –i
<interface>
```

2. Réglages de Snort

Tous les paramètres à régler de Snort se trouvent dans le fichier snort.conf situé dans le répertoire « etc ». Dans ce fichier une quantité importante d'information est fournie en commentaire pour permettre à l'utilisateur de connaître l'utilité de chaque section, de chaque variable et être capable de faire une configuration correcte.

Nous allons maintenant décrire les sections les plus importantes du fichier snort.conf.

réglage des variables réseau

La configuration d'un Système de Détection d'Intrusion sur un réseau dépend naturellement de la topologie de ce réseau, de ses plages d'adresses, des serveurs dont il est équipé, ... Par défaut, Snort va analyser tous les paquets de toutes les adresses IP du réseau. Ce choix de réglage se fait en utilisant la variable `homenet` réglée comme suit : `var $HOME_NET$ any`

Si l'utilisateur, ne veut analyser les paquets que de certaines adresses IP, il peut rentrer des intervalles d'adresses, des IP particulières, ...

les préprocesseurs

Les préprocesseurs ont été introduits dans la version 1.5 de Snort (et sont présents dans la version que nous utilisons). Les préprocesseurs permettent d'étendre les fonctionnalités de Snort en autorisant les utilisateurs et les programmeurs à ajouter ou à supprimer des modules.

D'un point de vue technique, le préprocesseur est exécuté avant que le moteur de détection ne soit appelé, mais après que le paquet ait été décodé. Les préprocesseurs sont chargés et configurés en utilisant le mot clé `preprocessor`. Le format de directive d'un préprocesseur dans une règle snort est de la forme:

```
preprocessor <name> : <options>
```

Les différentes options propres à chaque préprocesseur sont données dans le manuel à l'adresse suivante http://manual.snort.org/node17.html.

choix des règles

Cette section de snort.conf est une suite d' « include » de fichiers de règles. En effet, les règles pour un certain type d'attaques sont regroupées dans un même fichier au nom explicite avec l'extension « .rules » dans le répertoire « rules », par exemple : attack-responses.rules, ftp.rules, ... Pour que Snort utilise un fichier de règles, il suffit que cette ligne soit dé-commentée. Pour omettre un fichier de règles, il faut commenter la ligne avec un # devant.

Exercices :

Maintenant, vous allez tester deux préprocesseurs.

***Arpspoof** :*

1) *Via SCAPY, construire une requête arp pour demander l'@MAC de la machine d'à côté. Vérifier le résultat. Lire http://fr.wikipedia.org/wiki/Address_Resolution_Protocol*
2) *L'une des techniques pour écouter un trafic consiste à modifier le cache ARP d'une machine. Reproduisez ce type d'attaque avec SCAPY qui possède la fonction arpcachepoison. La syntaxe arpcachepoison(@IP de la machine à attaquer, @IP que vous voulez associer à votre @MAC). En analysant avec snort expliquez comment fonctionne cette attaque. http://en.wikipedia.org/wiki/ARP_spoofing*
3) *Reproduisez la même attaque sans utiliser la commande arpcachepoison(), c'est à dire juste avec des entités ethernet et arp. Il faut utiliser sendp() pour envoyer une trame.*
4) *Configurez le préprocesseur arpspoof pour qu'il détecte ce type de message. Testez avec scapy que snort détecte bien ce type de message.*
5) *Montrez les limites de cette approche dans un réseau constitué de machine linux en lisant la documentation du module ARP du noyau (utilisez la commande : man 7 arp)*

***Balayage :** L'une des techniques de balayage consiste à envoyer un segment tcp avec le bit SYN positionné à 1 sur une plage de ports (par exemple de 1 à 1024)*

1) *Déterminer le préprocesseur potentiel qui peut aider à la détection des balayages*
2) *Tester avec scapy en envoyant de tels segments avec comme adresse **destination votre PC**.*
3) *Quel est le nom de ce type de balayage ? Pour plus d'informations vous pouvez lire le chapitre 2 de « Secrets of Network Cartography: A Comprehensive Guide to nmap » (http://www.networkuptime.com/nmap/index.shtml)*
4) *Reproduire ce balayage avec nmap.*

3. Construction d'une règle

Dans cette section, nous expliquons la construction d'une règle snort. La syntaxe des règles Snort est relativement simple et flexible, et permet de construire des règles assez puissantes. Une règle s'écrit totalement sur une seule ligne (Snort ne gère pas les sauts de ligne) et elle se divise en 2 parties principales:

l'en-tête de règle qui contient comme informations

- l'action de la règle, c'est à dire la réaction de snort en fonction des paquets inspectés;
- le protocole qui est utilisé pour la transmission des données (snort en considère trois: TCP, UDP et ICMP);
- les adresses IP source et destination et leur masque;
- et les ports source et destination sur lesquels il faudra vérifier les paquets.

les options de la règle qui contient comme informations

- le message d'alerte;
- les informations permettant de dire si oui ou non il faut déclencher une alerte en fonction du paquet inspecté. Toutes les conditions permettant d'accepter ou de rejeter le paquet doivent être vraies pour que le paquet vérifie la règle et que l'action correspondante soit enclenchée.

3.1 L'en-tête de règle

C'est donc ici qu'on défini l'action à prendre et les paquets que l'on veut inspecter selon leur type, leur provenance et leur destination. On définit par action ce que Snort fera lorsqu'il découvrira un paquet qui correspond aux critères de la règle fournie. Il existe 5 actions dans Snort:

- alert : génère une alerte définie et journalise le paquet;
- log : journalise le paquet;
- pass : ignore le paquet;
- activate : fait une alerte et active une règle dynamique;
- dynamic : reste passif jusqu'à avoir été activé par une règle activate et puis agit comme une règle log.

On peut analyser 3 protocoles au moyen de Snort

- tcp;
- udp;
- icmp

On lui passe ensuite l'adresse ip ou une liste d'adresses ip avec un masque de sous-réseau si cela est nécessaire. On peut également utiliser un opérateur de négation « ! » qui dit à Snort de considérer toutes les adresses ip sauf celles-là. On peut également lui passer « any » qui dit de considérer toutes les adresses ip. Ensuite vient le port qui peut être défini précisément ou sur un intervalle. Par exemple « 1:1024 » veut dire que l'on scanne tous les ports de 1 à 1024. Si l'on utilise un numéro de port seul avec les « : » on spécifie de vérifier tout les ports plus grand ou égaux ou plus petit ou égaux au port donné selon que les « : » se trouvent devant ou derrière le numéro de port. Les opérateurs « ! » et « any » peuvent également être utilisés.
La direction du trafic auquel les règles s'appliquent est indiqué par un des opérateurs : « -> », « <- » ou « <> ». Le dernier étant l'opérateur bidirectionel.

3.2 Les options de la règle

Ce sont ces options qui sont le coeur même de la détection. Les diverses options peuvent être combinées à souhait pour former la règle la plus adéquate pour repérer les paquets qui nous intéressent. Deux options consécutives sont séparées par des ``;''. Voici une liste des options existantes dans Snort ainsi qu'un brève description.

- *msg* - affiche un message dans les alertes et journalise les paquets
- *logto* - journalise le paquet dans un fichier nommé par l'utilisateur au lieu de la sortie standard
- *content* - recherche un motif dans la charge d'un paquet
- *offset* - modifie l'option content, fixe le décalage du début de la tentative de correspondance de motif
- *depth* - modifie l'option content, fixe la profondeur maximale de recherche pour la tentative de correspondance de motif
- *nocase* - correspond à la procédure de chaîne de contenu sans sensibilité aux différences majuscules/minuscules
- session - affiche l'information de la couche applicative pour la session donnée
- *ip_proto* - permet de définir un protocole au dessus de IP
- *classtype* - assigne un classification à l'attaque (virus, cheval de troie, …)
- *flow* - permet de fixer le sens d'un flux, l'état de la connexion, …
- *react* - réponse active (bloque les sites web)

- …

Les options varient suivant les versions de snort. Plus d'informations sur le manuel de snort2.8.5 (il faut le lire pour connaître les paramètres des options)

3.3 Faux négatifs & Faux positifs

Les faux négatifs se produisent lorsque l'IDS ne détecte pas une attaque sur le réseau alors qu'elle est en train de se produire. Les faux positifs se produisent lorsque l'IDS signale une attaque sur le réseau alors qu'il s'agit d'une activité autorisée.

Les faux positifs diminuent la crédibilité que l'on peut avoir dans un IDS. En réglant Snort spécifiquement au réseau analysé, leur nombre peut être considérablement réduit (d'où l'importance de la configuration de Snort !!). Néanmoins, un IDS doit généralement générer un nombre raisonnable de faux positifs. S'il n'en génère aucun, il y a de bonnes chances que des faux négatifs se produisent et donc que des attaques réelles ne sont pas détectées. Il faut donc trouver un équilibre entre ces deux paramètres.

Exercices :
Commencer par mettre en commentaire ou supprimer toutes les lignes « include $RULE_PATH/* » (vous allez créer des règles, il ne faudrait pas qu'elles rentrent en conflit avec celles déjà définies dans snort). Créer votre fichier de règles et ajouter ce fichier dans snort.conf

Question 1 : Analyse de règles simples

1) Ajouter la règle suivante dans le fichier local.rules:

```
alert ip any any -> any any (msg:"BAD-TRAFFIC IP Proto 53 SWIPE"; ip_proto:53; reference:bugtraq,8211; reference:cve,2003-0567; classtype:non-standard-protocol; sid:2186; rev:3;).
```

Construire un paquet déclenchant la règle suivante et regarder l'alerte émise.

2) Retrouver pourquoi ce type de paquet est potentiellement dangereux. Que faut-il faire en tant qu'administrateur ?

Question 2 : Jouons avec la fragmentation

1) Refaites avec scapy le ping contenant « AAAABBBBCCCCCCC » de la question 6 des exercices de scapy.

2) Ecrivez une règle qui détecte les messages ICMP echo request contenant AAAABBBBCCCCCCCC. Vérifier qu'une alerte est bien générée avec scapy.

3) Changez la règle pour qu'elle émette une alerte lorsque que le message ICMP contient BBBBAAAACCCCCCCC.

3) Découpez le message ICMP en fragment IP de 8 octets. Modifiez le premier fragment (indice 0) pour qu'il contienne en plus de l'entête ICMP les données BBBBAAAA. Envoyez tous les fragments et vérifiez qu'une alerte est bien générée. Remarquez que le premier fragment a écrasé le deuxième au moment du réassemblage du paquet final.

4) Modifiez la politique de réassemble du préprocesseur frag3 par « last ». Refaites le test et vérifiez qu'aucune alerte n'a été générée.

Question 3 : Définition d'une règle pour snort

1) Installez un serveur FTP sur votre machine s'il n'en existe pas déjà. Pour cela suivez le tutoriel suivant http://blog.nicolargo.com/2009/01/un-serveur-ftp-en-5-minutes-chrono.html

2) Ecrire une règle snort qui émet une alerte lorsque trois demandes de connexion ont échoué en moins de deux minutes.

Question 4 : Installation d'un système de stockage et de visualisation d'alertes SNORT

Afin de faciliter l'analyse d'alertes SNORT, il est possible de stocker ces alertes dans une base de données et d'installer une interface pour les visualiser/faire des recherches/etc. Dans le cadre de ce TP, nous allons installer une solution où les alertes sont stockées dans une base de données MYSQL et l'interface web ACID BASE permet de visualiser les alertes.

1) Création de la base de données

```
aptitude install mysql-server mysql-client mysql-common
```

Pour cette installation, nous avons choisi le mot de passe « stri »

```
mysql -pstri
mysql> CREATE DATABASE snort;
mysql> grant CREATE, INSERT, SELECT, UPDATE on snort.* to snort@localhost;
mysql> grant CREATE, INSERT, SELECT, UPDATE on snort.* to snort;
mysql> SET PASSWORD FOR snort@localhost=PASSWORD('stri');
mysql> SET PASSWORD FOR snort=PASSWORD('stri');
mysql> flush privileges;
```

2) Installation de snort-mysql

```
aptitude install snort-mysql
```

Au moment de la configuration, garder le lancement de snort en mode manuel.

Vérification de la création des tables

```
mysql -pstri
mysql> show tables
```

Configuration de snort

```
dpkg --configure --pending
dpkg-reconfigure snort-mysql
```

Visualiser la configuration de snort dans /etc/snort/database.conf

3) Installation d'ACID BASE

```
aptitude install acidbase
```

Recharger la configuration du serveur Apache2

```
/etc/init.d/apache2 reload
```

4) Configuration d'acidbase

Se connecter sur le site http://localhost/acidbase
Cliquer sur le bouton comme indiqué
Lancer une alerte et vérifier sur la page d'acidbase qu'elle est bien stockée dans la base de données.

Question 5 : Jouons un peu plus avec la fragmentation.

L'objectif de cet exercice est de mieux comprendre la politique de réassemblage de fragments IP d'un système d'exploitation.

1) Créez un datagramme IP contenant
1111111122222222333333334444444455555555666666667777777788888888
2) Découpez ce datagramme en 4 fragments où :
- frag[0] contient 1111111122222222
- frag[1] contient 3333333344444444
- frag[2] contient 5555555566666666
- frag[3] contient 7777777788888888

2) Copiez frag[2] grâce à la fonction copy() dans un fragment de test. Modifiez les données du fragment de test pour qu'il contienne AAAAAAAABBBBBBBB.
3) Envoyez les fragments dans l'ordre suivant :
- frag[0], frag[1], frag[2], test et frag[3],
- puis frag[0], frag[1], test, frag[2] et frag[3].
Analysez les résultats obtenus
4) Modifiez le fragment de test pour que :
- son offset soit égal à 3
- ses données soient AAAAAAAABBBBBBBBCCCCCCCC
Envoyez les fragments dans l'ordre suivant :
- frag[0], frag[1], frag[2], test et frag[3],
- puis frag[0], frag[1], test, frag[2] et frag[3].
- puis frag[0], test, frag[1], frag[2] et frag[3].

Analysez le résultat et déduisez en la politique de réassemblage du système d'exploitation.

Pour plus d'information sur le chevauchement de fragment IP lire « IP Fragment Reassembly with scapy » à l'URL suivante : https://www.sans.org/reading-room/whitepapers/detection/ip-fragment-reassembly-scapy-33969

Question 6 : Exemple d'analyse : le cheval de Troie Phatbot

Phatbot est un cheval de Troie apparu au début des années 2000. Ce ver se propage grâce au protocole WASTE qui avait initialement conçu pour sécuriser et anonymiser les communications entre chateurs. Une alerte a été décrite par le CERTA (http://www.certa.ssi.gouv.fr/site/CERTA-2004-ALE-003/). Une description détaillée est proposée par F-Secure (http://www.f-secure.com/v-descs/agobot_fo.shtml)

Le groupe LURHP Threat Intelligence a analysé ce ver et a proposé les deux règles SNORT suivante :

```
alert tcp any any -> any any (msg:"Agobot/Phatbot Infection Successful";
flow:established; content:"221 Goodbye, have a good infection |3a 29 2e 0d
0a|"; dsize:40; classtype:trojan-activity;
reference:url,www.lurhq.com/phatbot.html; sid:1000075; rev:1;)

alert tcp any any -> any any (msg:"Phatbot P2P Control Connection";
flow:established; content:"Wonk-"; content:"|00|#waste|00|"; within:15;
classtype:trojan-activity; reference:url,www.lurhq.com/phatbot.html;
sid:1000076; rev:1;)
```

1) Montrer les limites de ces règles.
2) Une analyse plus poussée a été présentée par le SANS à l'URL suivante : http://pen-testing.sans.org/resources/papers/gcih/eradicating-masses-1-phatbot_106262.
Comment peut on détecter plus finement ce ver ? Est il possible d'écrire des règles SNORT pour émettre une alerte ? Quels autres outils peut on utiliser pour détecter l'activité de Phatbot ?

4. Pour aller plus loin dans la réflexion …

Et pour aller un peu plus loin que les sondes IDS telles que SNORT. Vous pouvez vous documenter sur les SIEMs (Security Information and Event Management) qui ont pour objectif de centraliser tous les évènements de sécurité afin de pouvoir appliquer des traitements de corrélation d'évènements poussés (Par exemple OSSIM : https://alienvault.bloomfire.com/posts/556521-a-beginner-s-guide-to-siem/public). Les SIEMs forment ainsi une brique nécessaire pour effectuer de la gestion de sécurité contextuelle en permettant de construire des informations de sécurité de haut niveau à partir de logs

divers provenant d'équipements réseau, d'applications ou encore de sondes IDS (Cf :http://www.sensage.com/sites/default/files/effective_security_monitoring_227893.pdf). Un exemple d'utilisation du SIEM OSSIM a été présenté à l'URL suivante (http://link.springer.com/content/pdf/10.1007%2F978-3-642-24270-0_15.pdf)