Mettre en place un IDS

Mario NGANGA
Groupe:2B
Groupe_SAE:06

Objectif : Détecter avec Snort les événements suivants :

- Tentative de connexion SSH sur les serveurs depuis l'extérieur
- Attaque DoS sur votre serveur Web avec des GET requests
- Détection des URIs non-normalisées
- Détection d'un login raté sur le serveur FTP
- Détection d'une attaque DoS avec TCP SYN
- Détection de paquets fragmentés de taille < 500 ou > 2000

Pour commencer la mise en place d'un IDS avec Snort, j'ai d'abord cherché à comprendre ce qu'est Snort, comment il fonctionne, comment construire des règles avec Snort, et ensuite j'ai commencé son installation sur la distribution Ubuntu.

Avant même de commencer à expliquer comment j'ai configuré Snort et la construction de règles pour atteindre les objectifs de la mise en place d'un IDS capable de détecter les six événements demandés, j'ai commencé par définir Snort et expliquer comment il fonctionne ainsi que son utilité.

Snort est le principal système de détection d'intrusions (IDS) open source au monde. Snort IDS utilise une série de règles qui aident à définir l'activité réseau malveillante et utilise ces règles pour trouver des paquets correspondants et générer des alertes pour les utilisateurs.

Configuration du Snort

Après l'installation de Snort,j'ai commencé à éditer le fichier **/usr/local/etc/snort/snort.lua**, qui est le fichier de configuration de Snort dans la distribution Ubuntu.

Dans ce fichier, j'ai modifié le champ **HOME_NET = '87.10.10.0/24'** pour représenter mon réseau local. Dans ce cas, l'adresse IP du réseau local est 87.10.10.0 avec un masque de sous-réseau /24.

Le champ **EXTERNAL_NET = '!$HOME_NET'** signifie que le réseau externe est tout ce qui n'appartient pas au réseau local défini par HOME_NET. Cela permet à Snort d'écouter le trafic

provenant d'autres réseaux que 87.10.10.0. En d'autres termes, Snort peut surveiller les réseaux autres que celui défini comme le réseau local.

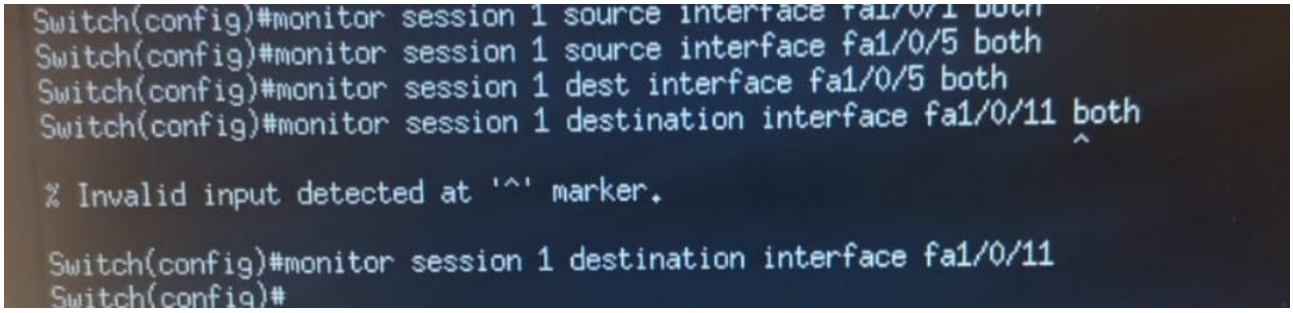
```
-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = '87.10.10.0/24'

-- set up the external network addresses.
-- (leave as "any" in most situations)
EXTERNAL_NET = '!$HOME_NET'
```

Ensuite, j'ai créé le fichier ***/usr/local/etc/rules/local.rules*** pour stocker toutes les règles me permettant de détecter les connexions SSH et FTP ratées, ainsi que d'autres règles pertinentes pour cet usage. Ce fichier contiendra les règles spécifiques à mon réseau ou aux menaces que je souhaite surveiller.

Ensuite, j'ai vérifié que toutes les règles étaient correctement configurées en utilisant la commande suivante :

snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/rules/local.rules

Configuration de la port Mirroring sur switch



Avec cette configuration, tout le trafic des ports fa1/0/5 (réseau A) et fa1/0/01 (réseau B) sera copié vers le port fa1/0/11 où se trouve votre IDS, lui permettant de détecter et d'analyser toutes les actions sur les réseaux A et B

Testes de Regles

J'ai commencé à faire les textes de 6 règles proposées pour cette SAE en tapant la commande :

snort -c /usr/local/etc/snort/snort.lua -R /usr/local/etc/rules/local.rules -i enxf8e43b5215d5 -A alert_fast -s 65535 -k none

Pour tester mes regles sur Snort fonctionnait avec le fichier de configuration et les règles spécifiés, en écoutant le trafic sur l'interface réseau *enxf8e43b5215d5*, en utilisant le mode d'alerte rapide, en analysant les paquets jusqu'à une taille maximale de 65535 octets et en désactivant le traitement des en-têtes TCP dupliqués.

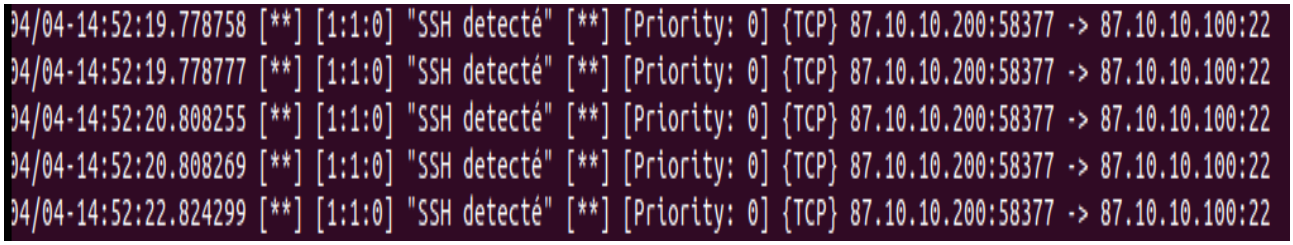
Tentative de connexion SSH sur serveurs depuis l’extérieur

Pour détecter les tentatives de connexion SSH depuis l'extérieur, j'ai créé la règle suivante :

```
alert tcp any any -> $HOME_NET 22 (
  msg:"SSH detecté";
  sid:1;
)
```

L'alerte est utilisée par Snort pour émettre une alerte après avoir détecté une connexion. TCP est effectivement le protocole, et le sens de la règle est de la source vers la destination.

Voici l'image du résultat du test de cette règle.



Cette image ci-dessus montre clairement que Snort parvient à détecter cette règle, mais il est nécessaire d'avoir d'abord une connexion SSH pour que Snort puisse détecter la tentative de connexion SSH.

Détection d'une (DoS) sur un serveur web à l'aide de requêtes GET.

Pour détecter les attaques (DoS) sur votre serveur Web à l'aide de requêtes GET, j'ai créé la règle suivante :

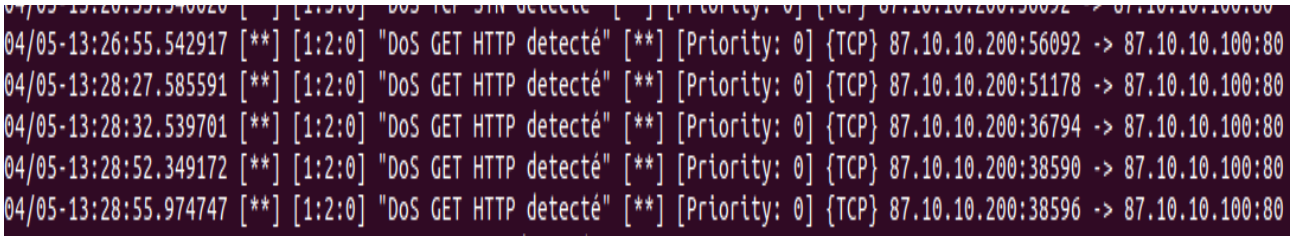
```
#DoS GET HTTP

alert tcp any any -> any 80 (
   msg: "DoS GET HTTP detecté";
   http_method;
   content:"GET";
   sid:2;
)
```

Pour tester cette règle, j'ai utilisé deux méthodes :

1- Depuis le client B, je me suis connecté au serveur Web et j'ai rafraîchi la page.

2- J'ai utilisé l'outil curl pour envoyer des requêtes HTTP au serveur :

curl 87.10.10.100



Détection d’un login raté sur le serveur FTP

Pour la détection le login FTP raté, j'ai créé la règle suivante :

```
#Login raté FTP
alert tcp any any -> any any (
    msg:"FTP Login raté";
    service:ftp;
    content:"530",nocase;
    sid:8;
)
```

Pour tester j’ai essayé de me connecter au serveur FTP depuis le client B avec un faux login.

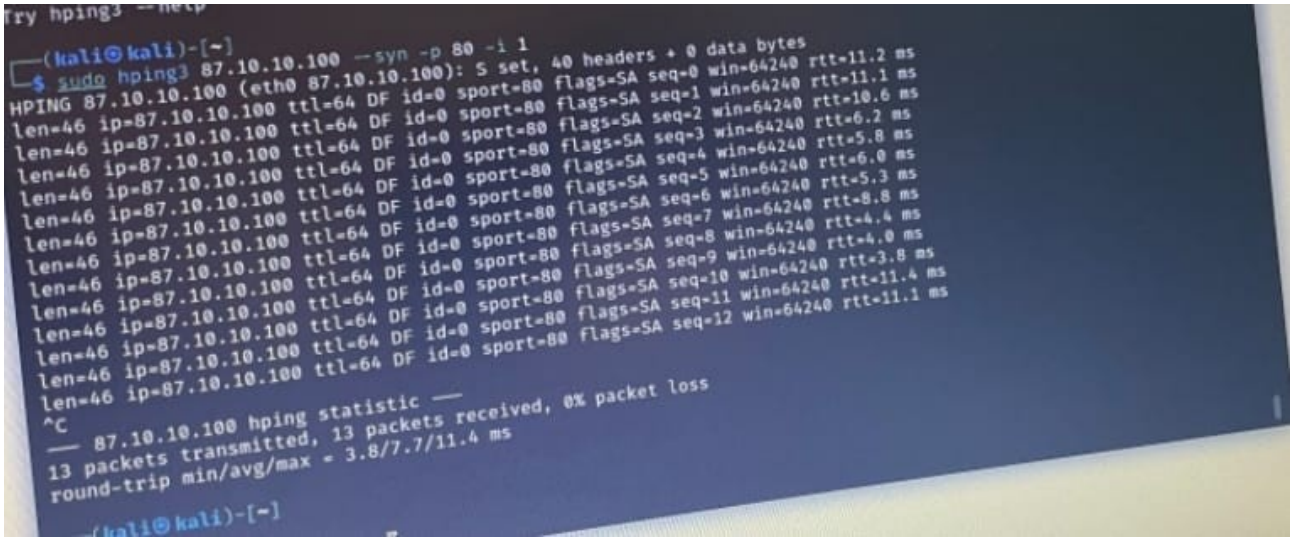
```
04/05-13:03:02.360694 [**] [1:8:0] "FTP Login raté" [**] [Priority: 0] {TCP} 87.10.10.100:21 -> 87.10.10.200:42188
04/05-13:03:46.196161 [**] [1:8:0] "FTP Login raté" [**] [Priority: 0] {TCP} 87.10.10.100:21 -> 87.10.10.200:35308
04/05-13:04:08.406005 [**] [1:8:0] "FTP Login raté" [**] [Priority: 0] {TCP} 87.10.10.100:21 -> 87.10.10.200:35154
```

Détection d’une attaque DoS avec TCP SYN

Pour detecter la ataque DOS avec TCP SYN j’ai crée la regle suivant :

```
#DoS TCP SYN
alert tcp any any -> $HOME_NET any (
    msg:"DoS TCP SYN detecté";
    flags:S;
    detection_filter:track by_dst, count 5, seconds 10;
    sid:5;
)
```

Pour tester cette règle, j'ai lancé une attaque DOS TCP SYN depuis le client à l'aide de l'outil hping3.



Après l'attaque, Snort a bien détecté le type d'attaque. Voici l'image ci-dessous: