TP 1 - Prise de contact avec Snort, scapy

0. Initialisation du TP

Installer les paquets python-scapy, snort, nmap.

1. Présentation de SNORT v2.8.5

La détection d'intrusion consiste en un ensemble de techniques et méthodes utilisées pour détecter des activités suspectes au niveau d'un réseau et /ou d'un équipement terminal. Les systèmes de détection d'intrusion sont classés en deux catégories : les systèmes basant leur analyse sur des signatures et les systèmes détectant des anomalies. La première catégorie utilise le fait qu'une intrusion possède une signature (ports particuliers, mots clés dans les données utiles,…) de la même façon que les virus. Le système de détection d'intrusion basé sur les signatures tente alors de trouver ces signes particuliers dans les paquets examinés. Les systèmes de la seconde catégorie détectent les anomalies dans les entêtes des paquets par rapport aux protocoles standards.

SNORT est un système de détection d'intrusion réseau (Network Intrusion Detection System) open source couramment utilisé. Il permet d'analyser les flux de données par rapport à une base de données de signature, mais aussi de détecter les anomalies.

1.1 Architecture de SNORT

SNORT est constitué de plusieurs composants logiques. Ces composants permettent de détecter une attaque particulière et de générer en sortie une alerte dans le format désiré. Les composants principaux de SNORT sont : le décodeur de paquet, les préprocesseurs, le moteur de détection, le système de journalisation et d'alerte et les modules de sortie.

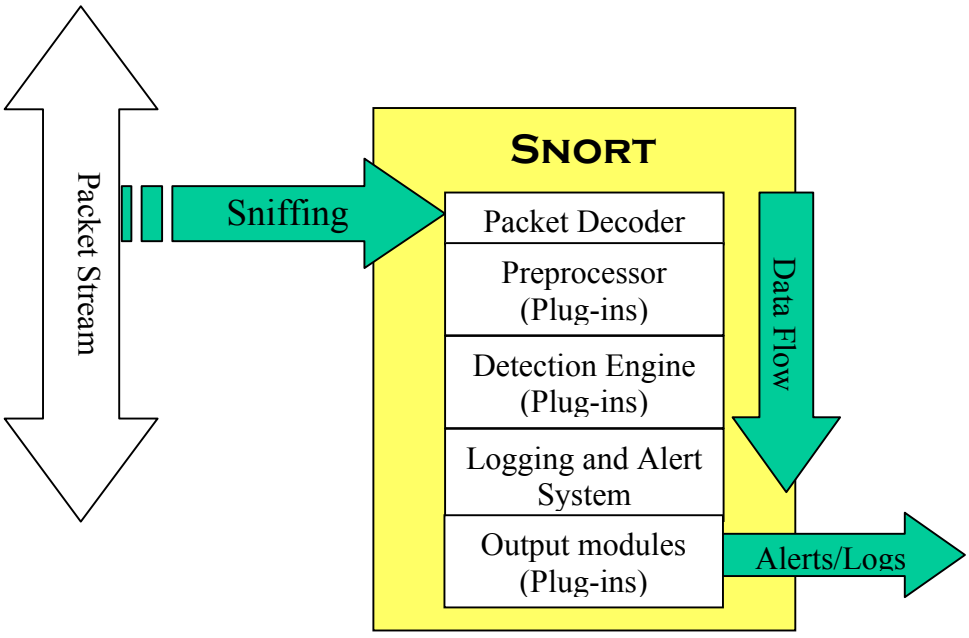

Figure 1. Architecture de SNORT

1.1.1 Le décodeur de paquet

Le décodeur de paquet récupère les paquets provenant de différents types d'interface réseau et le prépare pour les préprocesseurs ou le moteur de détection. Les interfaces réseaux peuvent être de type Ethernet, SLIP, PPP, etc.

1.1.2 Les préprocesseurs

Les préprocesseurs sont des modules d'extension pour arranger ou modifier les paquets de données avant que le moteur de détection n'intervienne. Certains préprocesseurs détectent aussi des anomalies dans les entêtes des paquets et génèrent alors des alertes.

La préparation des paquets pour le moteur de détection est très importante dans le cadre des IDS. En effet, les attaquants utilisent différentes techniques pour duper les IDS. Par exemple, si vous créez une règle pour retrouver la signature « scripts/iisadmin » dans les paquets HTTP. Le moteur de détection cherchant exactement cette chaîne de caractère peut être dupé par un attaquant qui modifie légèrement la chaîne de caractère. Par exemple : « scripts/./iisadmin », « scripts\iisadmin », « scripts/.\iisadmin ».

Pour compliquer un peu la tâche, le attaquant peut aussi insérer dans l'URI (Uniform Resource Identifier) des caractères hexadécimaux ou unicodes qui sont légaux. Le préprocesseur peut réarranger la chaîne de caractères afin d'être détectable par l'IDS.

Les préprocesseurs sont aussi utilisés pour le réassemblage de fragments IP. Lorsqu'une quantité importante de données sont transférées, les paquets sont souvent fragmentés. Par exemple, la longueur maximale des paquets sur un réseau de type Ethernet ne peut excéder 1500 octets (valeur définie par le MTU). Ceci implique qu'un paquet d'une longueur supérieure est découpé en plusieurs paquets de longueur égale ou inférieure à 1500 octets. Le système terminal qui réceptionne les données est capable de réassembler le paquet original. Pour pouvoir détecter des signatures, un IDS positionné sur un équipement intermédiaire doit faire de même afin de déjouer les attaques des attaquants qui utilisent cette technique de fragmentation.

Les préprocesseurs inclus SNORT permettent de défragmenter les paquets, de décoder les URI HTTP, de réassembler les flux TCP, etc. Néanmoins, il est possible d'écrire son propre préprocesseur (cf README.PLUGINS du répertoire snort/doc).

1.1.2.1 Frag3

Le préprocesseur frag2 permet à SNORT de contrer les attaques de type fragmentation IP. Ce type d'attaque consiste à fragmenter les paquets IP afin de cacher certaines informations. Par exemple, il est possible par ce type d'attaque de segmenter l'information TCP sur plusieurs paquets (attaque Tiny Fragments, 8 premiers octets de TCP dans le premier paquet IP, i.e. ports source et destination + numéro de séquence, et le reste dans les second paquet IP, i.e. flags syn/ack/fin) ou de falsifier des informations en jouant sur l'offset (attaque Fragment Overlapping, premier paquet IP avec flag syn =0 et second paquet avec les informations de connexion qui écrase les flags TCP avec syn=1 et ack=0).

1.1.2.2 Stream5

Le préprocesseur Stream5 offre des mécanismes pour surveiller des sessions TCP et UDP. Il analyse aussi les messages ICMP pour vérifier les messages d'inaccessibilité et d'indisponibilité. Ce préprocesseur permet en autre :

- le réassemblage de flux TCP d'une même session. Si un attaquant essaye d'attaquer un système en introduisant des éléments dans les données utiles de plusieurs paquets TCP d'une session Telnet, ce module en assemblant ces données utiles permet de retrouver la signature.
- et l'inspection stateful. Ce module permet de garder en mémoire l'état d'une connexion TCP.

Ainsi, ce préprocesseur est capable de détecter des irrégularités dans les flux TCP. Par exemple, la méthode « TCP stealth » consiste à ne jamais achever complètement le processus d'établissement de la connexion TCP.

1.1.2.3 sfPortscan

La première étape d'un processus d'attaque consiste le plus souvent à déterminer les services fonctionnant sur un réseau à l'aide d'outils tel que nmap. Une fois cette étape effectuée, des attaques sur les vulnérabilités connues des services détectés sont lancées. Ce préprocesseur permet de détecter les balayages d'adresses IP et de ports TCP/UDP. Différents types de balayages sont concernés : Un -> plusieurs hôtes (recherche d'un service particulier), un -> plusieurs ports (recherche des services sur un équipement).

1.1.2.4 http_inspect

Un grand nombre d'attaque sur les serveurs WWW sont véhiculées en utilisant des caractères hexadécimaux dans les URI. Par exemple, en considérant la règle SNORT qui doit reconnaître la chaîne de caractère « /wwwboard/passwd.txt », un attaquant peut la déjouer en envoyant la requête pour obtenir l'URI « %2Fwwwboard%2Fpasswd.txt ». Le préprocesseur http_inspect normalise les requêtes afin de permettre leur analyse par le moteur de détection.

1.1.2.5 Smtp

Ce préprocesseur permet d'analyser les messages SMTP.

1.1.2.6 ssh

Le préprocesseur ssh permet de détecter les exploits suivants Challenge-Response Buffer Overflow, CRC 32, Secure CRT, and the Protocol Mismatch.

1.1.2.7 ssl

Le préprocesseur ssl permet d'analyser la phase de négociation de SSL.

1.1.2.8 ftp_telnet

Ce préprocesseur travaille uniquement sur les protocoles telnet et ftp. Il décode ou supprime les codes de contrôle telnet binaires insérés de façon aléatoire dans un flux telnet ou ftp. Cette technique est souvent utilisée par les pirates pour exécuter des commandes système via, par exemple une connexion ftp.

1.1.2.9 rpc_decode

Le préprocesseur rpc_decode normalise les connexions RPC. Les attaquants exploitent ce protocole pour faire de la reconnaissance ou de l'exploitation à distance en établissant des liaisons dynamiques avec les services distants.

1.1.2.10 arpspoof

Le protocole ARP (Address Resolution Protocol) permet de déterminer l'adresse MAC associée à une adresse IP. Ce protocole est à la base de plusieurs attaques pour écouter (par exemple le paquetage dsniff) ou encore « spoofer » (par exemple, rediriger le trafic d'un hôte vers une autre destination).

Le préprocesseur arpspoof détecte les anomalies dans les messages ARP, par exemple :

- Pour les requêtes ARP : si l'adresse MAC source Ethernet et celle contenue dans le message ARP sont différentes, si les adresses MAC source ne correspond pas à l'adresse IP
- Pour les réponses ARP : les adresses MAC dans la trame Ethernet et dans le message ARP sont différentes
- Les requêtes unicast.

1.1.3 Les moteur de détection

Le moteur de détection de SNORT constitue le cœur de l'IDS. Il est responsable de détecter toute activité d'intrusion dans les flux de données. Il utilise des règles qui consistent à une définition d'un ensemble de critères sur les paquets. Si un paquet correspond à une règle, une action est réalisée (journaliser, déclenchement d'une alerte).

Selon la puissance de l'équipement où est installé SNORT et le nombre de règles définies, le nombre de paquets analysés peut être plus ou moins important.

1.1.4 Le système de journalisation et d'alerte

Selon les décisions prises par le moteur de détection, les paquets peuvent être journalisés ou générer une alerte. La journalisation peut s'effectuer par de simples fichiers textes, des fichiers selon le format tcpdump, ou selon d'autres formes.

1.1.5 Les modules de sortie

Les modules de sortie peuvent effectuer différentes opérations selon la manière dont on désire sauvegarder les informations générées par le système de journalisation et d'alerte :

- Enregistrement simple dans un fichier (comme /log/snort/alerts),
- Envoyer des notifications d'évènement SNMP,
- Enregistrer dans une base de données comme MySQL,
- Transformer dans un format XML,
- Envoyer des messages SMB (Server Message Block) dans le cas de machines Windows,
- Mèl, SMS, etc.

2. Utilisation de SNORT comme un simple sniffeur

Dans le mode sniffeur réseau, SNORT capture et affiche les paquets selon différents niveaux de détails. Aucun fichier de configuration n'est requis.

2.1. Affichage à l'écran

snort -v : Imprimer les informations contenues dans les entêtes TCP/IP
snort -dv : Imprimer les informations contenues dans les entêtes IP/UDP/TCP/ICMP
snort -edv : Imprimer les informations des entêtes MAC
snort -edv proto : Imprime uniquement les messages du protocole PROTO (i.e. ip, tcp, udp, icmp)
snort -h : aide
l'option –I permet de spécifier l'interface d'écoute
Ctrl-C : Arrêter SNORT

Exercice :
1) Tester les commandes et retrouver les différents champs des entêtes en lançant différentes applications (www,ftp,mail…).
2) Limiter l'affichage des flux de données reçus à un protocole particulier.

2.2 Enregistrement dans un fichier

2.2.1 Enregistrement en format texte

Il possible d'enregistrer les paquets dans des fichiers en mode texte en ajoutant l'option « –l <nom du répertoire> ». Créez un répertoire log dans votre répertoire courant.

snort -dev -l ./log -K ascii : Enregistre les affichages dans des fichiers du répertoire « log »

Plusieurs répertoires sont créés dans le répertoire cible. Chacun de ces répertoires correspond à un hôte et contient plusieurs fichiers. Chaque répertoire est identifié par l'adresse IP de l'hôte associé. Les fichiers contiennent les données concernant les différentes connexions et les différents types de données réseau.

2.2.2 Enregistrement en format binaire

Sur les réseaux haut débit, le fait d'enregistrer les données en format ASCII dans un nombre important de répertoires différents peut entraîner une charge du système très importante. SNORT peut enregistrer les données dans un fichier binaire au format tcpdump (sniffeur sur UNIX), et de les regarder ensuite.

snort -l ./log -b : Enregistrer les données dans un fichier binaire
snort -dev -r /temp/snort.log.xxxxxxx : Lire les données dans un fichier au format binaire

Exercice :
3) Regarder comment SNORT enregistre les données au format texte.
4) Même chose pour le format binaire. Utiliser la commande pour lire les fichiers binaires.
5) Quel est l'avantage du format binaire par apport le format texte.

2.3. SCAPY

Scapy est un (fantastique) outil de manipulation de paquet interactif écrit en python. Il est capable, entre autres, d'intercepter le trafic sur un segment réseau, de générer des paquets dans un nombre important de protocoles, de réaliser une prise d'empreinte de la pile TCP/IP, de faire un traceroute, d'analyser le réseau informatique, etc. Scapy peut être utilisé via un interpréteur de commande grâce à la commande scapy ou à travers des scripts python.

Il va nous permettre d'effectuer des tests d'intrusion dans la suite des TP.

La documentation de scapy se trouve à l'adresse suivante : http://www.secdev.org/projects/scapy/doc/usage.html#interactive-tutorial. Il est possible d'avoir une documentation sur les différentes fonctions scapy dans l'interpréteur de commande grâce à la fonction **help()** (par exemple, help(IP))**.** Pour avoir l'ensemble des champs d'une entête d'un protocole, il suffit d'afficher l'attribut **fields_desc** (par exemple IP.fields_desc).

Par exemple, les commandes suivantes permettent de créer un datagramme IP avec un TTL à 123 et contenant comme données utiles « BONJOUR » à destination de 127.0.0.1 et de l'envoyer sur le réseau :

```
>>>ip=IP()
>>>ip.ttl=123
>>>ip.dst= « 127.0.0.1 »
>>>ip.payload=« BONJOUR »
>>>ip
<IP ttl=123 dst=127.0.0.1 |<Raw load='BONJOUR'|>>
>>>send(ip)
.
sent 1 packets.
```

Une autre manière pour effectuer la même chose dans un script :

```
#!/usr/bin/python
from scapy.all import *

ip=IP(dst="127.0.0.1", ttl=123)/"BONJOUR"
send(ip)
```

Exercice :
6) Lancez SNORT pour qu'il affiche les messages ICMP. Lancez SCAPY en tant qu'interpréteur de commande. Recréez avec SCAPY la commande ping. Le message ICMP généré contiendra comme données « AAAABBBBCCCCCCCC ». Vérifiez avec Snort que le message est bien formulé en regardant la réponse au ping.
*7) Grâce à la fonction **fragment()**, découpez votre ping en fragments de 8 octets. Envoyez les fragments et vérifiez que la réponse au ping est correcte.*

8) Il est possible de manipuler des paquets réseaux via des scripts. Pour cela créez un fichier testping.py. Afin de pouvoir utiliser SCAPY comme une bibliothèque python ajoutez au début du fichier les lignes suivantes :

```
#!/usr/bin/python
from scapy.all import *
```

A la suite, ajouter les instructions SCAPY nécessaire pour créer le ping de la question 6. Lancez le script testping.py et vérifiez le bon fonctionnement.

9) Modifiez le script pour que :

a) vous envoyez un message ICMP echo request

b) lorsque vous recevez le message ICMP echo reply, vous récupérez le champ id (noté id_{reply}) de l'entête IP.

c) vous envoyez un nouveau message echo request de telle manière que le champ id de l'entête IP soit égal à $id_{reply}+1$.

*Vous utiliserez la fonction **sr1()** pour envoyer les message ICMP echo request afin d'obtenir la réponse associée. Pour visionner les messages, vous utiliserez la fonction **show2()**.*