NET5531: TP - Détection d'intrusion et corrélation d'alertes

François-Xavier Aguessy
fx@thetreep.com

21 octobre 2019

Objectifs

L'objectif de ce TP est de mettre en oeuvre les connaissances théoriques vues dans le cours de détection d'intrusion et corrélation d'alertes. Ainsi, nous allons apprendre à configurer et utiliser des outils de détection d'intrusion. Puis, nous écrirons des règles de détection d'intrusion que nous visualiserons dans des interfaces dédiées. Enfin, nous utiliserons un outil de corrélation pour tirer profit au mieux des alertes reçues.

1 Prise en main des outils

1.1 Machine virtuelle

Le TP se déroulera sur une machine virtuelle reliée au réseau. Celle-ci est configurée sur la distribution open source SELKS [1], basée sur Debian. La documentation de cette distribution est principalement disponible sur le wiki du dépôt github du projet : `https://github.com/StamusNetworks/SELKS/wiki`.

La machine virtuelle doit être lancée de préférence avec au moins 4Gb de RAM et 2 vCPU. L'interface réseau doit être configurée en *promiscuous mode*.

Info: Les identifiants pour se connecter à la machine virtuelle sont :
utilisateur : `selks-user`
mot de passe : `selks-user`

1.2 Principaux outils

Voici les principaux outils installés sur la machine virtuelle, ainsi que des liens vers leur documentation :

Suricata : `https://suricata.readthedocs.io/`

Scirius : `https://scirius.readthedocs.io`

Evebox : `https://evebox.readthedocs.io`

Kibana : `https://www.elastic.co/guide/en/kibana/current/`

Elasticsearch : `https://www.elastic.co/guide/en/elasticsearch/reference/current/`

Logstash : `https://www.elastic.co/guide/en/logstash/current/`

Dsiem : `https://github.com/defenxor/dsiem/tree/master/docs`

[1] https://www.stamus-networks.com/open-source/#selks

2 Écriture de règles Suricata

La première partie de ce TP va consister à écrire des règles pour l'IDS Suricata. Le plus simple pour charger des règles sur Suricata dans SELKS est d'écrire les règles dans un fichier de règles (par exemple le fichier `tp.rules` situé sur le Bureau), puis de le charger par l'intermédiaire de Scirius dans la collection de règles préconfigurée pour le TP : `https://127.0.0.1/rules/source/3/edit`. Puis, il faut recharger les règles de la sonde Suricata sur `https://127.0.0.1/suricata/update`

Info: Attention à bien remplir le `sid` des nouvelles règles (avec un numéro > à 1.000.000) et incrémenter leur numéro de version `rev` pour que les nouvelles règles soient prises en compte.

La syntaxe pour écrire des règles Suricata est rétrocompatible avec celle de Snort, avec quelques subtilités. Ne pas hésiter à se reporter à la documentation : `https://suricata.readthedocs.io/en/suricata-5.0.0/rules/index.html`

2.1 Règle 1 : accès au serveur HTTP

Le but de cette première règle est de détecter dès qu'il y a un accès au serveur HTTP du TP. Un prérequis sera de connaître l'adresse IP et le port TCP utilisés par le serveur HTTP de test du TP. Il est conseillé d'utiliser par exemple Whireshark pour cela.

Écrire la règle Suricata permettant de détecter tous les accès à ce serveur. La charger sur la sonde Suricata et vérifier qu'elle est fonctionnelle dans l'interface d'EveBox ou de Kibana.

2.2 Règle 2 : accès avec succès à une page d'administration

Sur le serveur HTTP, il y a une page d'administration accessible après authentification simple HTTP avec la requête suivante : `GET /admin/private.txt`. Ajouter une règle à la configuration de Suricata pour détecter les accès réussis à cette page.

Info: Pour tester l'efficacité de l'alerte, il est possible de se connecter à la page d'administration avec les identifiants suivants

utilisateur : `admin`

mot de passe : `verysecure`

2.3 Règle 3 : accès avec erreur d'authentification

Ajouter une troisième règle à la configuration de Suricata pour détecter les erreurs d'authentification à cette même page.

2.4 Règle 4 : détection d'une attaque par dictionnaire

Nous allons désormais écrire une règle pour détecter une vraie attaque possible : une attaque par dictionnaire : un attaquant essaye plusieurs mots de passe successifs issus d'un dictionnaire de mot de passe, pour se connecter. Le but est donc de détecter un enchainement de quelques requêtes d'authentification erronées en peu de temps.

Info:

Pour tester, il est possible de simuler cette attaque à l'aide d'un petit client HTTP fourni sur la machine virtuelle en exécutant dans un terminal

Command Line

```
$ ./httpclient -server X.X.X.X:1234 -dictionary
```

3 Corrélation d'alertes

Le but de cette deuxième partie est d'apprendre à utiliser les outils de visualisation des alertes fournis dans la distribution SELKS, et à écrire des règles simples de corrélation, en utilisant le logiciel open source `dsiem`.

3.1 Détection graphique d'une attaque par force brute

L'objectif de cette partie est de créer un graphe dans Kibana et de l'intégrer dans un Dashboard de votre choix, pour permettre de détecter une attaque par force brute. Une attaque de ce type se manifestera par un grand nombre d'accès avec erreur d'authentification. Il faut donc créer une visualisation permettant d'afficher le volume d'alertes en fonction du temps. Lors du TP, une attaque par force brute sera lancée sur le serveur HTTP, à vous d'identifier quand elle s'est produite.

3.2 Détection d'un scénario d'attaque

La dernière partie de ce TP consiste à utiliser le logiciel open source `dsiem` pour détecter des scénarios d'attaque plus complexes, combinant plusieurs alertes. Le scénario à détecter est le suivant : une attaque par dictionnaire est effectuée et l'attaquant finit par aboutir au vrai mot de passe et donc arrive à récupérer les données confidentielles du serveur.

Pour cette partie, à vous de jouer, en vous basant sur la documentation de `dsiem` pour essayer d'écrire la règle de corrélation permettant de détecter ce scénario d'attaque réussi, tout en réduisant le plus possible les faux positifs, malgré le trafic (simulé) qui circule sur le réseau.

L'attaque sera lancée en fin de TP, quels seront les binômes qui arriveront à la détecter ?

Info:

Pour vous aider, l'outil `dsiem` a été installé et configuré pour récupérer par Logstash les alertes de Suricata et les alarmes générées par les règles de corrélation sont stockées dans Elasticsearch pour être affichées soit dans Kibana, soit dans l'interface dédiée de `dsiem` accessible à `http://127.0.0.1:8080/ui/`.

Les règles de corrélation peuvent être écrites dans `/var/dsiem/configs` et chargées en redémarrant le service avec

Command Line

```
$ sudo service dsiem restart
```