YNOV CAMPUS

Mise en place de Suricata

Sécurité des SI

Mickael Rigonnaux & Fabio Pace
12/04/2019

Table des matières

1. Présentation du TP

Ce TP a été réalisé dans le cadre du cours de sécurité des SI au sein du Campus Ynov d'Aix-en-Provence. Il a pour but de procéder à l'installation du NIDS (Network Intrusion Detected System) Suricata au sein du réseau du campus par des membres du BSI.

2. Présentation de Suricata

Suricata est un IDS/IPS basé sur des signatures, c'est un outil Open Source sous licence GPL v2. Le projet a été créé de 0 en 2008 par Victor Julien mais il est maintenant porté par l'OISF (Open Information Security Fondation). La fondation s'occupe principalement de trouver des financements pour ce projet qui est aujourd'hui utilisé par de nombreuses entreprises.

La 1ère version stable de Suricata a été publiée en 2010, cette 1ère version avait pour objectif de mettre à disposition un moteur d'IDS/IPS multithread supportant les règles de Snort. Aujourd'hui la dernière version stable de Suricata est la 4.1.3.

3. Contexte

Pour ce TP l'IDS est installé au sein de l'infrastructure du campus Ynov d'Aix-en-Provence. L'infrastructure est principalement utilisée via le wifi par des étudiants, on compte à peu près 500 à 600 étudiants sur le campus. La partie réseau de l'infrastructure est gérée principalement par du matériel Cisco Meraki.

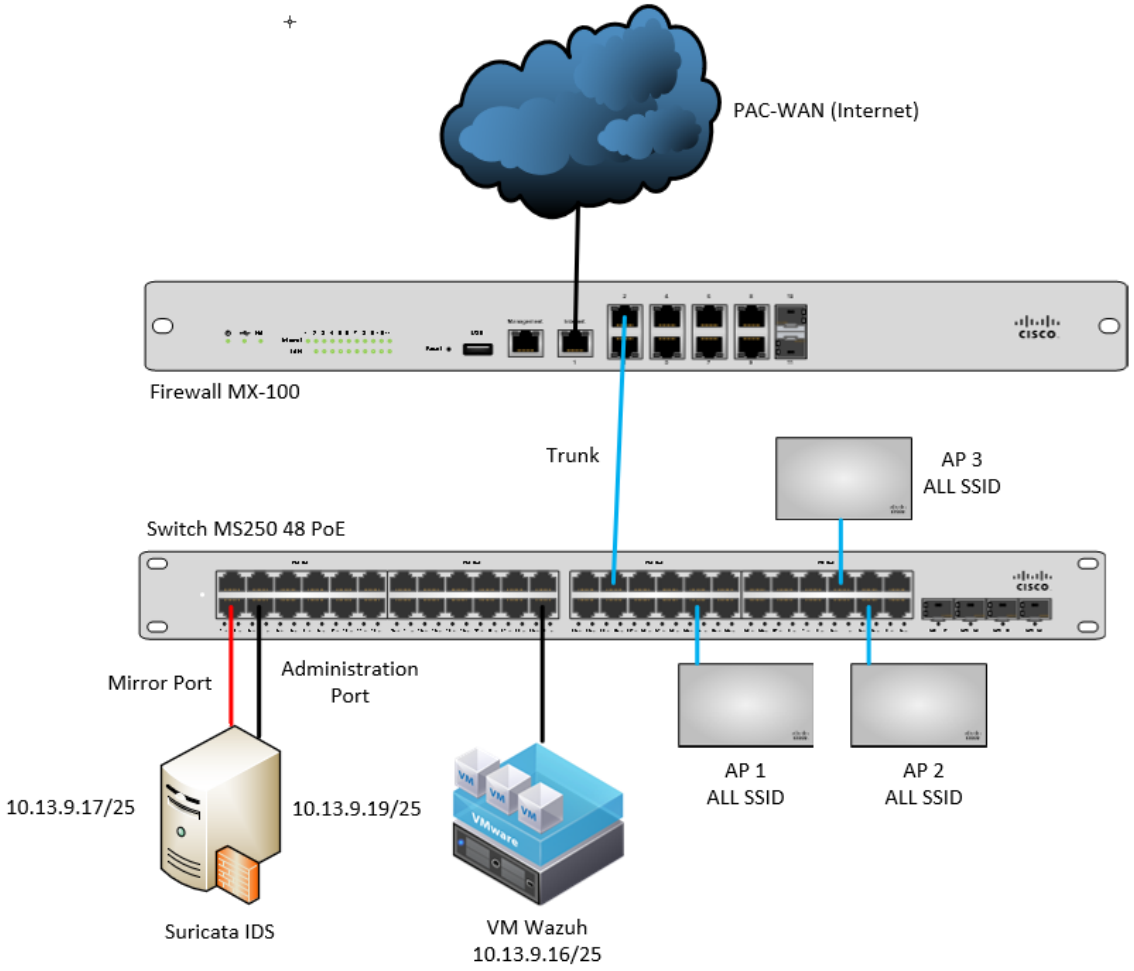
Le campus abrité également une administration et des professeurs ainsi qu'un parc de serveur virtualisé avec des ESXi.

Dans un premier temps et pour garantir la sécurité des étudiants nous avons donc choisi de mettre en place l'IDS seulement sur la partie wifi de ce campus. Pour ce faire nous avons branchés directement Suricata au cœur de réseau avec deux interfaces. Une première pour l'administration de la machine et une seconde pour la récupération des flux.

Dans ce même temps nous choisissons de mettre en œuvre seulement la partie IDS de l'outil et non pas l'IPS (Intrusion Prevention System). Car nous ne connaissons pas encore assez bien le fonctionnement du réseau pour réaliser des actions automatiquement.

Pour des raisons de simplicités et de performances évidente la machine Suricata est une machine physique.

4. L'infrastructure



Dans notre cas la machine Suricata à ses deux interfaces dans le réseau serveur. L'un des deux interfaces sert uniquement pour la récupération des flux des 12 bornes wifi. Sur ces bornes wiki plusieurs SSID et réseaux sont présents filtrés par un Radius. L'ensemble des équipements réseaux sont dans un réseau à part pour l'administration, il ne sera pas traité dans ce document.

Description de la machine

Nom	OS	IP	Réseau	Description
Suricata	Centos 7	10.13.9.19	10.13.9.0/25 (SRV)	Interface Management
		10.13.9.17		Interface sondes AP
Wazuh (VM)	Centos 7	10.13.9.16		----------

Au niveau des caractéristiques sur la machine Suricata :

- Intel i5 4440
- 16 Go de RAM DDR3
- 2 cartes réseaux Gigabit
- 500 Go de stockage (HDD)

5. IDS & Suricata

5.1. Introduction

Un IDS dans notre cas peut avoir un rôle crucial pour la sécurité des étudiants, il permettra de détecter et d'alerter quasiment en temps réel les membres du BSI en cas d'attaque sur le réseau. Cependant un IDS comme tous les systèmes ne protège pas complètement des attaques le réseau, il y a généralement un grand nombre de faux positifs avec ce genre d'outil. Il faut donc ajouter des règles au fur et à mesure et établir d les évènements à remonter ou non afin de pouvoir générer correctement les alertes.

Notre choix s'est donc tourné vers Suricata pour plusieurs raisons :

- Il est plus « puissant » que Snort car il introduit le multithreading et du coup il offre théoriquement la possibilité de traiter davantage de règles sur l'IDS plus rapidement et donc de traiter un volume plus important sur le même matériel
- Il intègre également un langage de script Lua qui offre une plus grande flexibilité pour créer des règles en identifiant des évènements plus difficiles voire impossible à détecter avec les règles classiques

Cependant, Suricata n'a pas que des avantages, il est plus compliqué à installer que Snort et il ne gère pas les requêtes ARP par exemple car il n'a pas de préprocesseur.

Nous devons donc trouver une alternative pour gérer ce cas. Des outils tels que ARPWatch, NetCut… ne permet de récupérer les requêtes ARP seulement du réseau de la machine. Nous avons donc trouvé trois fork d'ARPWatch permettant d'outre passer la notion de VLAN et de pouvoir récupérer les requêtes ARP de tous les réseaux :

- https://github.com/fln/addrwatch
- https://github.com/verbnetworks/arpwitch
- https://github.com/SgtMalicious/Arpwatch-NG-VLAN

5.2. Installation de Suricata IDS

Pour l'installation, nous vous invitons à lire l'article que nous avons écrit sur le blog net-security.fr :

https://net-security.fr/security/installation-dun-nids-suricata/

5.3. Attaque ARP

Nous avons souhaité d'utiliser l'outil « addrwatch » pour la documentation et la robustesse du fork :

```
addrwatch enp2s0
```

```
1554969506 enp2s0 0 00:0c:29:43:5a:d9 10.13.9.2 ARP_REQ
1554969506 enp2s0 0 00:0c:29:43:5a:d9 10.13.9.2 ARP_REQ
1554969507 enp2s0 0 00:0c:29:43:5a:d9 10.13.9.2 ARP_REQ
1554969509 enp2s0 0 00:11:32:14:a9:c8 10.13.9.55 ARP_REP
1554969509 enp2s0 0 00:11:32:14:a9:c8 10.13.9.55 ARP_REP
1554969512 enp2s0 0 00:1d:09:20:ff:62 10.13.9.1 ARP_REQ
```

- ARP_REQ : Paquet ARP que la machine envoie pour mettre à jour les tables ARP (toutes les x minutes ou secondes)
- ARP_REP : Packet ARP pour répondre à une demande « A qui appartient cette adresse MAC ? »

Nous nous sommes rendu compte que nous récupérons seulement les requêtes ARP du VLAN de production.

Nous avons donc utilisé tcpdump pour récupérer tout le trafic ARP et vérifier les différents paquets que Méraki envoi grâce au port miroir, et nous nous apercevons que nous récupérons seulement des requêtes ARP du vlan de production :

```
09:11:51.973993 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:03.973953 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:03.973983 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:08.590514 ARP, Request who-has 10.13.9.110 (Broadcast) tell 10.13.9.110, length 46
09:12:22.073943 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:22.073974 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:34.073922 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:34.073945 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:35.689859 ARP, Request who-has 10.13.9.83 tell gateway, length 46
09:12:38.640424 ARP, Request who-has 10.13.9.110 (Broadcast) tell 10.13.9.110, length 46
09:12:40.196063 ARP, Request who-has 10.13.9.21 tell srvad.ynovaix.com, length 46
09:12:40.196501 ARP, Request who-has 10.13.9.21 tell srvad2.ynovaix.com, length 46
09:12:40.604601 ARP, Reply gateway is-at 0c:8d:db:fe:de:ff (oui Unknown), length 46
09:12:52.073928 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:52.073960 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:12:57.538373 ARP, Request who-has veeam.ynovaix.com tell srvad.ynovaix.com, length 46
09:13:01.375058 ARP, Request who-has lab-vrops.ynovaix.com tell srvad.ynovaix.com, length 46
09:13:04.173908 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:13:04.173939 ARP, Reply sauvegarde.ynovaix.com is-at 00:11:32:14:a9:c8 (oui Unknown), length 46
09:13:08.690555 ARP, Request who-has 10.13.9.110 (Broadcast) tell 10.13.9.110, length 46
```

La requête ARP encadré en rouge est la requête type que nous devons récupérer pour vérifier si une attaque d'ARP spoofing est en place.

Nous pouvons également créer notre propre solution de détection d'ARP Spoof :

```
tcpdump -i enp2s0 -F "dump.arp" | grep "ARP, Reply gateway is-at"
```

Nous vérifions si l'adresse MAC est différente de celle qui est stocké en base (celle de Meraki par exemple).

Méraki ne permet pas d'envoyer les requêtes ARP des autres VLANs sur l'interface du suricata.

Nous recherchons donc activement comment pouvoir détecter cette attaque au sein du réseau d'Ynov Aix.

5.4. Attaque scan NMAP

Nous avons implémenté la règle suivante pour détecter tout scan NMAP effectués par les utilisateurs du réseau :

```
alert tcp any any -> $HOME_NET any (msg:"SCAN synscan portscan"; id: 39426; flags:
SF;reference:arachnids,441; classtype:attempted-recon; sid:630; rev:1;)
```

Si nous souhaitons peaufiner nos règles pour la détection des scans NMAP :

- Pour une détection de la version du SSH :

```
alert tcp any any -> $HOME_NET 22 (msg:"SCAN SSH Version map attempt";
flow:to_server,established; content:"Version_Mapper"; nocase;
classtype:network-scan; sid:1638; rev:4;)
```

- Pour une détection de l'UPNP dans le réseau :

```
alert udp any any -> $HOME_NET 1900 (msg:"SCAN UPNP service discover
attempt"; content:"M-SEARCH "; offset:0; depth:9; content:"ssdp\:discover";
classtype:network-scan; sid:1917; rev:3;)
```

- Pour une détection du scan NMAP via l'User-Agent NMAP :

```
alert tcp any any -> $HOME_NET $HTTP_PORTS (sid:xxx; gid:1; flow:established,to_server;
content:"Nmap"; http_header; fast_pattern:only;
pcre:"/^User\x2dAgent\x3a\x20[^\r\n]*Nmap/Hm"; metadata:service http;
reference:url,https://nmap.org/book/nse.html; msg:"BLACKLIST User-Agent known malicious user-
agent string Nmap - scanner"; classtype:network-scan; rev:2; )
```

Grâce au User-Agent, nous pouvons détecter beaucoup d'outils d'attaque comme par exemple une attaque SQL via SQLMap :

```
alert tcp any any -> $HOME_NET $HTTP_PORTS (sid:xxx; gid:1; flow:established,to_server;
content:"sqlmap"; http_header; fast_pattern:only;
pcre:"/^User\x2dAgent\x3a\x20[^\r\n]*sqlmap/Hm"; metadata:service http;
reference:url,http://sqlmap.org/; msg:"BLACKLIST User-Agent known malicious user-agent string
sqlmap - vulnerability scanner"; classtype:network-scan; rev:1; )
```

5.5. Attaque ICMP Smurf

L'attaque ICMP smurf permet de réaliser un DOS (Deny of service) sur une machine victime via ne inondation de requête ICMP via une "armée" de machine et une requête en direction du broadcast.

La solution afin d'éviter une attaque ayant une grande ampleur est de minimiser les plages réseaux pour éviter d'y avoir beaucoup de machine qui puisse répondre à une demande broadcast. Une autre solution peut-être également de bloquer les requêtes ICMP vers le broadcast, ces configurations la sont possible sur du matériel comme Meraki.

Pour détecter une attaque ICMP via Suricata, il suffit d'ajouter la règle suivante :

```
alert icmp any any -> $HOME_NET any (msg:"GPL SCAN Broadscan Smurf Scanner";
dsize:4;icmp_id:0; icmp_seq:0; itype:8; classtype:attempted-recon; sid:2100478; rev:4;)
```

6. L'alerting

Suricata nous permet donc de détecter les intrusions que nous avons définis précédemment.

Nous utilisons Wazuh afin de récupérer les alertes de Suricata et d'envoyer des alertes par mails aux membres du BSI car il est serveur de mail dans l'infrastructure du campus. De plus, cela permet d'avoir une gestion centralisée des alertes.

Nous utilisons donc Wazuh (HIDS) pour emmètre les alertes sur le campus. Nous installons donc l'agent Wazuh sur Suricata en ajoutant la configuration suivante :

```
<localfile>
  <log_format>json</log_format>
  <location>/var/log/Suricata/eve.json</location>
</localfile>
```

Cette configuration permet à Wazuh de lire le fichier de log en temps réel. Nous indiquons également que le fichier est en JSon et l'emplacement de ce même fichier.

Il faut savoir que par défaut Wazuh possède des règles pour Suricata : https://github.com/wazuh/wazuh/blob/master/etc/rules/0475-suricata_rules.xml

Dont la règle suivante qui permet d'avoir une alerte lorsque Suricata rencontre une signature :

```
<group                                                    name="ids,suricata,">
<rule id="86601" level="3">
        <if_sid>86600</if_sid>
        <field name="event_type">^alert$</field>
        <description>Suricata: Alert - $(alert.signature)</description>
</rule>
</group>
```