

Rapport de projet

Systèmes de détection d'intrusion

Date de rendu	17 janvier 2014
Rédigé par	Julien Bourdon, Julien Legras et Jean-Baptiste Souchal
À l'attention de	Djelloul Ziadi

Table des matières

1	Présentation générale des systèmes de détection d'intrusion	3
2	Snort	5
2.1	Introduction	5
2.2	Fonctionnalités	5
2.3	Fonctionnalités avancées	6
3	Suricata	8
3.1	Introduction	8
3.2	Fonctionnalités	8
3.3	Fonctionnalités avancées	9
4	Snorby	11
5	Prélude	12
5.1	Introduction	12
5.2	Fonctionnalités	13
5.3	À propos	13
6	Démonstration de Snort	14
6.1	Scan de ports	14
6.2	Injection SQL dans l'URL	15
6.3	Injection XSS	15
6.4	Erreurs 403	15
6.5	Fuite de /etc/passwd	15
7	Ressources	15

1 Présentation générale des systèmes de détection d'intrusion

Un IDS (Intrusion Detection System) a pour but de surveiller, contrôler et détecter les attaques potentielles sur un réseau ou une machine. Ce système est essentiellement composé d'un sniffer¹ associé à un moteur qui analyse le trafic selon des règles pré-établies dans des fichiers de configuration. En effet, ce système de détection est, de nos jours, devenu indispensable au sein d'une architecture informatique sécurisée.

L'IDS peut analyser trois couches :

- couche Réseau (IP, ICMP)
- couche Transport (TCP, UDP)
- couche Application (HTTP, Telnet)

Il existe deux types d'IDS :

Le HIDS (Host IDS) qui est directement placé sur un ordinateur hôte pour surveiller le système et ses applications. Ce type de système est très utile pour s'assurer qu'une machine n'est pas infectée.

Le NIDS (Network IDS) qui est une sonde placée sur le réseau pour surveiller ce dernier dans son ensemble. Il capture tout le trafic et l'analyse pour rechercher les paquets suspects.

Le problème réside dans le fait que certains paquets considérés comme suspects ne le sont en fait pas : les *faux-positifs*. À l'inverse, des paquets malveillants peuvent passer sans être détectés par la sonde, ce sont des *faux-négatifs*. Ces faux-négatifs utilisent une technique appelée Evasion pour ne pas être repéré par la sonde.

Selon le type de trafic analysé, l'IDS peut faire différentes actions :

- journaliser l'événement (source d'information et vision des menaces courantes)
- avertir un système avec un message (appel SNMP)
- avertir un humain avec un message (courrier électronique, SMS, interface web, etc.)
- amorcer certaines actions sur un réseau ou hôte (mettre fin à une connexion réseau, ralentir le débit des connexions, etc.)

Les méthodes de détection se distinguent en deux grandes familles : par signature ou comportementale.

La méthode de détection par signature est la plus simple car elle est basée sur la reconnaissance de schémas connus par expressions régulières. Chaque attaque connue a sa signature associée stockée dans une base de données. Ces signatures sont comparées systématiquement à celles des paquets qui transitent. On a donc peu de faux-positifs mais il faut forcément que l'attaque soit connue pour être détectée. Cela implique une maintenance régulière de la base de données. La grosse faiblesse est que l'utilisation de techniques d'évasion incapacite totalement ce type de défense.

La famille comportementale utilise deux méthodes différentes : la détection d'anomalie ou la vérification d'intégrité.

Pour la détection d'anomalie, on considère qu'un comportement qui diffère du comportement "normal" du système est suspect. Une attaque inconnue peut tout de même être détectée grâce à ce mécanisme. Pour décrire un comportement normal, il faut créer un profil. Ce profil a plusieurs caractéristiques (volume des échanges réseaux, appels systèmes, commandes usuelles, etc.) et repose sur plusieurs outils assez complexes. Cette méthode est difficile à tromper, permet la détection d'attaques jusqu'alors inconnues et donc de créer de nouvelles règles pour les contrer. Mais malheureusement, le taux de faux-positifs est trop important et la mise en place d'un profil est longue et difficile.

Pour la vérification d'intégrité, le principe est de générer une somme de contrôle sur les fichiers du système. Cette somme est ensuite comparée à un instant T pour vérifier qu'il n'y a pas eu de modifications importantes. C'est une des méthodes les plus employées car elle est simple et efficace.

Pour résumer, les IDS sont évidemment devenus indispensables mais il y a encore de nombreux points négatifs. En effet, c'est une technologie complexe qui nécessite un degré d'expertise assez élevé. De plus, pour le rendre le plus optimisé possible, il faut beaucoup de temps. On peut dire que les IDS sont promis à un avenir radieux mais que cet avenir n'est pas encore arrivé.

Un IPS (Intrusion Prevention System) est un IDS à qui on ajoute des fonctionnalités de blocage, on l'appelle aussi IDS actif. Son but peut être d'interrompre ou de ralentir une connexion mais aussi de blacklister les sources dangereuses grâce à un firewall et à un proxy.

Le souci qui peut provenir d'un IPS est qu'un faux-positif va être bloqué immédiatement et peut donc paralyser le réseau.

On peut placer un IDS/IPS à plusieurs endroits où leur rôle diffère :

Entre internet et le firewall (Honey Pot) C'est une machine ou un programme volontairement vulnérable destiné à attirer et à piéger les pirates qui sert à occuper et garder la trace de ce pirate mais qui donne également des informations sur de nouvelles attaques. En effet, vu la quantité de trafic qui passe par lui, il serait quasiment impossible de logger toutes les informations qui transitent.

Entre le firewall et la DMZ Il permet de détecter les attaques non-filtrées par le firewall, le log de tout ce qui passe peut être réalisé sans problème et donne donc des informations claires.

Entre le firewall et le réseau interne L'un des plus importants car on sait qu'environ 80% des attaques proviennent de l'intérieur (trojans, virus...).

¹Un sniffer est un logiciel capable de capturer, lire et enregistrer les paquets qui transitent sur le réseau.

2 Snort



2.1 Introduction

Pour ce projet, nous avons décidé d'utiliser Snort car c'est un des plus utilisé avec Suricata, qu'il peut faire IDS/IPS et aussi parce qu'il est installé par défaut sur netkit (réseau virtuel que nous allons utiliser pour simuler les attaques). Snort est un IDS open source conçu en 1998 par Marty Roesch qui a été racheté par la suite par SourceFire. C'est le logiciel le plus utilisé (plus de 2 millions de téléchargements) et il est très souvent mis-à-jour.

Il peut interagir avec le firewall pour bloquer des intrusions (mode IPS) à l'aide de différents plugins. Bien évidemment, il est paramétrable et on peut donc ajouter des règles nous-mêmes. Le petit bémol est qu'il ne gère pas l'envoi de mails ou de SMS pour prévenir d'éventuelles attaques. Pour ce faire, on peut bien sûr utiliser d'autres logiciels en complément.

2.2 Fonctionnalités

Snort se décompose en plusieurs blocs :

- le décodeur de paquets, qui récupère des paquets de différentes interfaces réseau et prépare les paquets pour passer dans le préprocesseur.
- les préprocesseurs sont des composants qui peuvent modifier ou défragmenter les paquets de données avant que le moteur de détection fasse ses opérations pour découvrir si le paquet est utilisé par un intrus.
- le moteur de détection qui utilise l'algorithme AHO-CORASICK.
- le système d'alerte et d'enregistrement qui sert à prévenir et à logger les informations sur l'attaque.
- les modules de sortie qui permettent d'envoyer tous les logs vers une base de données.

Voici un exemple de règle pour détecter une connexion avec le login root sur le port ftp :

```
alert tcp any any -> 192.168.1.0/24 21 (content: "USER root";  
                                     nocase;  
                                     msg: "FTP root user access attempt";)
```

Et voici un exemple d'une autre pour une connexion à un site non-autorisé :

```
alert tcp any any <> 192.168.1.0/24 any (content-list:"adults";  
                                     msg: "Adults list access attempt";  
                                     react: block;)
```

Les règles commencent toujours par l'action qui va être lancée.

En mode IDS, il y a 5 actions par défaut :

- alert : génère une alerte puis log le paquet.

- log : log seulement le paquet.
- pass : ignore le paquet et le laisse passer.
- activate : génère une alerte et active une règle dynamique.
- dynamic : une fois activée par une règle de type activate, elle se comporte comme un log. Cela peut être très utile quand on veut récupérer le contenu d'un buffer-overflow pour voir quelles données sensibles ont été touchées.

En ajoutant le mode inline de Snort (IPS), on ajoute ces 3 règles de protection :

- drop : bloque et log le paquet.
- reject : en plus de faire ce que les règles drop font, elle envoie un "TCP reset" pour les connexions TCP ou un "ICMP port unreachable" pour les connexions UDP.
- sdrops : bloque le paquet mais ne le log pas.

On peut également créer nos propres types de règles qu'on pourra utiliser par la suite en tant qu'actions :

```
1 ruletype redalert
2 {
3   type alert
4     output alert_syslog: LOG_AUTH LOG_ALERT
5     output log_tcpdump: suspicious.log
6 }
```

Ce nouveau type enverra des logs dans le syslog ainsi que sur tcpdump.

Ensuite, le deuxième champ spécifie le protocole, pour le moment on peut utiliser TCP, UDP, ICMP et IP. Dans des versions futures, il est possible qu'on voit apparaître ARP, IGRP, GRE, OSPF, RIP, IPX, etc. ...

Ensuite, viennent les adresses IP (source puis destination). Le mot-clé "any" prendra toutes les adresses en considération. Par contre, une adresse unique devra être spécifiée comme ceci : "192.168.1.0/24". Pour mettre une liste d'adresses, on les met entre crochets et elles sont séparées par une virgule sans espaces.

Au contraire, si l'on veut que tout le trafic à l'exception d'une adresse soit contrôlé, on pourra utiliser le caractère "!" devant l'adresse.

Pour les ports qui suivent les deux adresses, il existe également le mot-clé "any". On peut également créer une sorte de "range" comme ceci (tout dépend du placement du caractère " : ") :

- 1:1024 = les ports contrôlés iront de 1 à 1024.
- :1024 = les ports contrôlés seront inférieurs ou égaux à 1024.
- 1024: = les ports contrôlés seront supérieurs ou égaux à 1024.

Enfin pour finir, les deux adresses IP sont séparées par un opérateur. Le plus souvent il s'agit de "->" qui signifie que le trafic contrôlé ira de la source (à gauche) vers la destination (à droite). Mais il existe aussi l'opérateur "<>" qui signifie que le trafic contrôlé sera dans les deux sens tant qu'il est entre ces deux adresses. Il n'existe pas d'opérateur "<-" car il est inutile et pourrait amener des erreurs.

2.3 Fonctionnalités avancées

Il existe quatre catégories principales pour les options de règles :

- general : donne simplement de l'information.

- payload : recherche dans les paquets qui ont une charge utile.
- non-payload : recherche dans les paquets qui n'ont pas de charge utile.
- post-detection : lance une autre règle à la suite d'une règle.

Pour les règles contenues dans "general" on a principalement :

- msg : sert à afficher un texte spécifique.
- reference : utilisé avec d'autres plugins pour plus d'information sur une attaque.
- gid : qui récupère l'id du générateur de Snort .
- sid : récupère l'id d'une règle.
- classtype : une attaque est rattachée à une classe d'attaque et cette classe est renvoyée.
- priority : chaque classe d'attaque a une priorité associée qu'on peut également redéfinir (high,medium,low)

Quelques exemples pour les règles "payload" :

- content : Si le paquet contient les mots recherchés, alors il est repéré. Cette méthode est la plus utilisée et la plus efficace si elle est bien comprise.
 - nocase : Enlève le casse, souvent associé à "content".
 - depth : Spécifie à quel niveau de profondeur Snort doit aller pour chercher les informations.
 - offset : Spécifie à partir de quel endroit dans un pattern déjà défini, la recherche doit commencer.
- ↔ Il existe en tout 45 règles qui sont principalement utilisées avec Content...

Pour les "non-payload", on ne va pas rentrer dans le détail non plus mais voici quelques exemples :

- ttl : compare le temps de trajet d'un paquet pour voir par où il est passé.
 - id : pour repérer l'id d'une adresse IP (31337 très utilisé par les hackers)
 - flags : chercher si un flag précis est présent dans le paquet (FIN, SYN, RST...)
- ↔ Il existe en tout 22 règles pour ce type.

Pour les "post-detection" il n'y a pas beaucoup de règles mais elles sont importantes :

- logto : envoie toutes les données dans un fichier spécifique (utile pour l'analyse).
- resp : qui kill la session d'où provient l'attaque.
- react : affiche une page web à l'attaquant et ferme la connexion.
- tag : ne garde pas que le paquet, mais log aussi les IP et host source/destination.
- detection-filter : en fonction de certains critères, l'action n'est lancée qu'au moment où tous ces critères ont été remplis (nombre de connexions ratées à la suite. . .)

Il existe un système d'activation/désactivation d'autres règles au bout d'un certains nombres de détections/protections.

3 Suricata



3.1 Introduction

Suricata est un IDS/IPS open source soutenu par The Open Information Security Foundation (OISF). Il est compatible avec Snort mais est également son concurrent direct. Voici une liste des fonctionnalités qui diffèrent avec Snort :

Suricata	Snort
Soutenu par une fondation	Développé par SourceFire
Multithreadé	Multiprocesseur
IPS natif	IPS supporté
Fonctions avancées (flowint, libHTTP)	jeu de règles SO (performant mais fermé)
Support de PF_RING	Pas d'accélération matérielle
Code moderne et modulaire	Code vieillissant
Jeune mais dynamique	10 ans d'expérience

↔ Comparaison plus complète : <http://www.aldeid.com/index.php/Suricata-vs-snort>

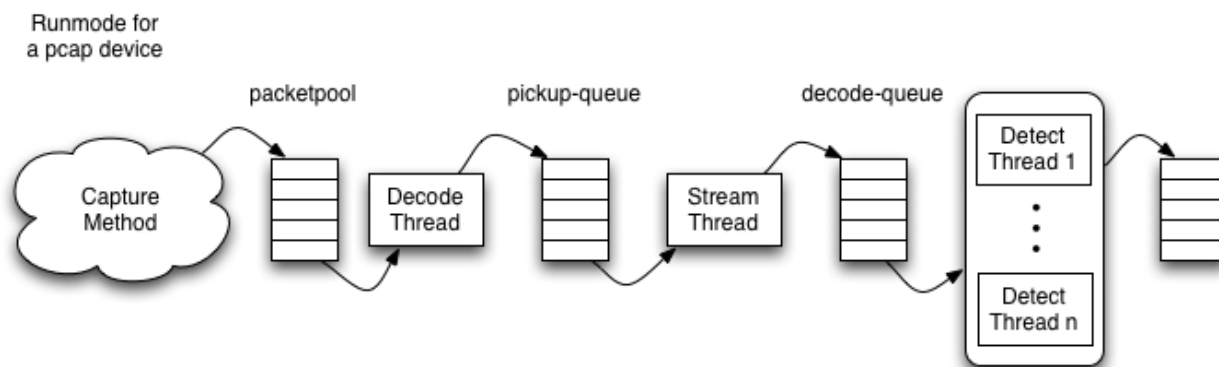
PF_RING est une bibliothèque réseau open source permettant de capturer un flux réseau sur un lien à haut débit (>1Gbps). Une fois ce trafic capturé, il est possible de l'analyser et/ou le manipuler.

3.2 Fonctionnalités

Du fait de son jeune développement, Suricata est axé sur des fonctionnalités modernes telles que :

- support natif de l'IPv6
- multi-threadé
- accélération matérielle native (GPU, PF_RING)
- IPS natif

Son architecture se décompose en une suite de modules de traitement. Chaque suite de traitement peut avoir sa propre architecture. Voici l'architecture en mode pcap automatique :



Suricata permet également un paramétrage fin des préférences CPU telles que :

- affectation d'un thread à un CPU
- affectation d'une famille de threads à un ensemble de CPU
- prise en compte des interruptions matérielles

Il existe différents modules d'entrées pour Suricata, qu'il soit en mode IDS ou IPS :

IDS	IPS
PCAP (live, multiinterfaces, hors-ligne)	NFQueue (cible iptables)
PF_RING	ipfw : pare-feu avec états pour systèmes BSD
AF_PACKET	

En modules de sortie, Suricata nous propose :

- fastlog
- unified log (Barnyard 1 & 2, format utilisé par Snort)
- HTTP log (log format Apache)
- Prelude

Au niveau des signatures, Suricata supporte la quasi-totalité des signatures de Snort mais également des fonctionnalités exclusives utilisées par les rulesets comme VRT (Sourcefire Vulnerability Research Team) ou Emerging Threats.

Le support de l'accélération matérielle par Suricata s'effectue avec CUDA, l'architecture de calcul parallèle développée par NVIDIA. Actuellement, Suricata a implémenté un algorithme de matching en CUDA.

3.3 Fonctionnalités avancées

Suricata utilise la bibliothèque libHTTP qui est un parseur orienté sécurité du protocole HTTP. Cette bibliothèque permet de faire du suivi de flux et est capable de décoder des flux compressés par GZip. Voici un exemple de règle pour un chat Facebook :

```

1
2 alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS \

```

```
3 (
4 msg: "ET CHAT Facebook Chat (send message) " ; \
5 flow : established,to_server ; content : "POST" ; http_method ; \
6 content : \
7 "/ajax/chat/send.php" ; http_uri; content : "facebook.com" ; http_header ; \
8 classtype : policy-violation; reference : url, doc.emergingthreats.net/2010784; \
9 reference : \
10 url, www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/POLICY/POLICY_Facebook_Chat; \
11 sid :2010784; rev: 4 ; \
12 )
```

Cette signature teste :

- la méthode HTTP : POST
- la page : /ajax/chat/send.php
- le domaine : facebook.com

Suricata gère les variables de flux également, cela permet de détecter les attaques par étapes et de créer une machine à état au sein du flux (automate). Il existe deux types de variables de flux :

Flowbits	Flowint
condition booléenne	définition de compteur
positionnement d'un drapeau	opération arithmétique

Voici un exemple qui remonte une alerte si et seulement si usernamecount est plus grand que 5 :

```
1 alert tcp any any -> any any (msg: "Counting Usernames" ; content : "jonkman" ; \
2 flowint: usernamecount , + , 1 ; flowint: usernamecount , > , 5 ;)
```

Suricata supporte l'extraction et l'inspection de fichiers (uniquement transmis par HTTP) à l'aide de différents mots clefs :

- filemagic : description du contenu ("executable for MS Windows", "JPEG image data", "RSA private key"...)
- filestore : stockage du fichier pour inspection
- fileext : extension du fichier
- filename : nom du fichier

Une autre fonctionnalité intéressante est l'analyse de handshake TLS. Plusieurs mots clefs sont supportés :

- TLS.version : Correspondance avec le numéro de version du protocole
- TLS.subject : Correspondance sur le sujet du certificat
- TLS.issuerdn : Correspondance sur le nom du générateur de certificat

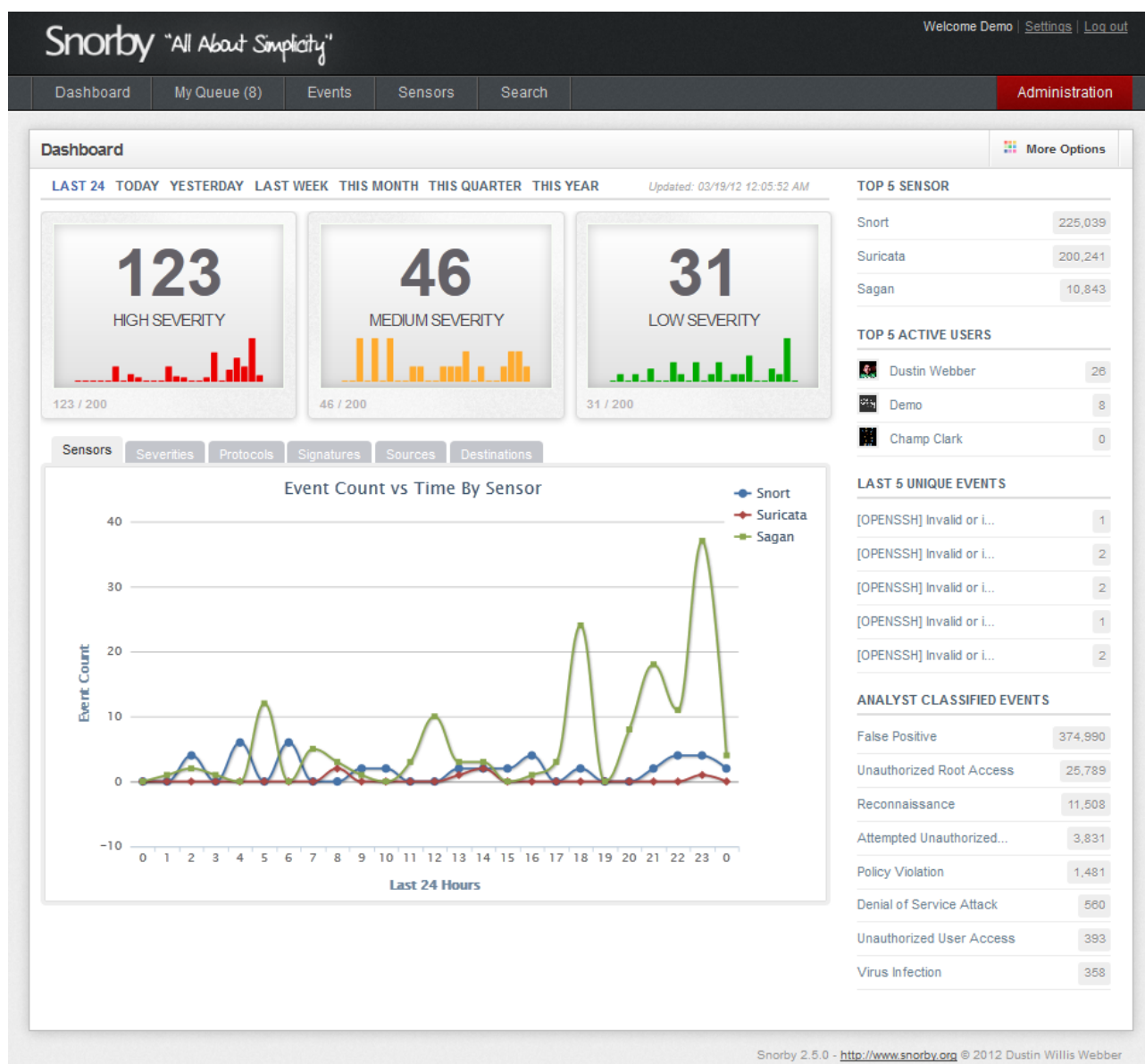
On peut ainsi vérifier que les employés d'une entreprise utilisent bien des certificats fournis par la PKI de l'entreprise.

4 Snorby

Snorby est une application qui apporte une interface web afin de faire du monitoring sur les logs générés par les IDS comme Snort ou Suricata. La gestion des logs devient alors beaucoup plus simple, rapide et plus efficace. Cette application permet entre autre de :

- visualiser en temps réel les détections des outils IDS (snort, suricata)
- afficher le détail de chaque log (visualiser l'adresse IP source, le contenu des paquets ...)
- générer des diagrammes sur les types d'attaque, les fréquences de chaque attaque ...
- générer des PDF, exporter des données (fichier csv par exemple)
- ajouter des règles de classification sur les détections
- envoyer des notifications par mail (voir directement sur le smartphone via l'application iOS)

A noter que Snorby est 100% Open Source et est en permanence maintenu et développé par une équipe professionnelle. Snorby est à l'origine une création Dustin Webber ; mais est maintenant attachée à l'entreprise "Threat Stack" dont Dustin Webber est le co-fondateur.

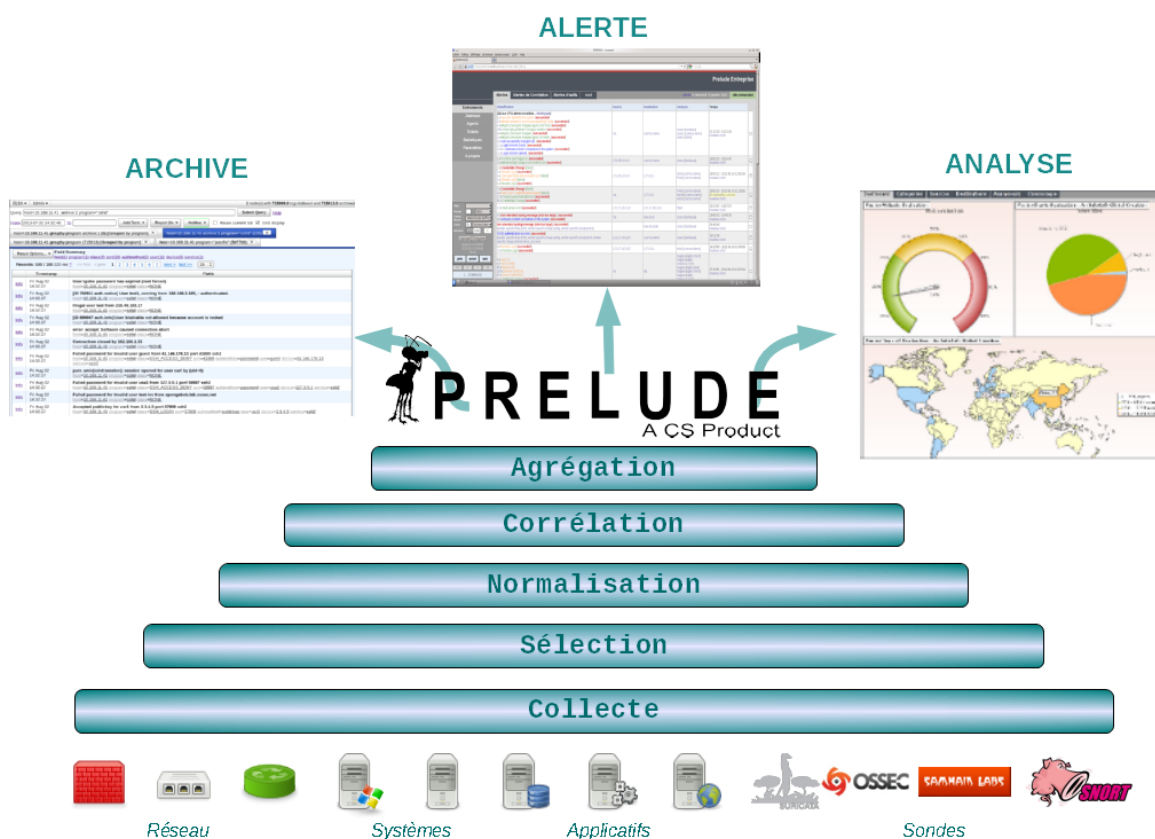


5 Prélude



5.1 Introduction

Prelude est un SIEM (Security Information and Event Management) qui a pour rôle de gérer les événements du système d'informations. C'est donc une solution de supervision de sécurité. Prelude collecte, filtre, normalise, corrèle, stocke et archive tous les événements de sécurité indépendamment des types d'équipements surveillés issus du système d'informations. Au-delà de sa capacité de traitement de tous types de journaux d'événements (logs système, syslog, fichiers, etc.), Prelude est nativement compatible avec de nombreuses sondes anti-intrusion. Les informations collectées par Prelude sont donc récupérées au préalable par le système, le réseau ou des sondes comme Snort et Suricata par exemple. On parle de corrélation car ces solutions sont munies de moteurs de corrélation qui permettent de relier plusieurs événements à une même cause. Grâce à cette corrélation, certains événements système ou réseaux, qui détectés séparément auraient générés qu'une alerte mineur, peuvent être mis en relation et de ce fait générer une alerte majeure pour la sécurité du système d'informations. Donc à partir de l'ensemble de ces informations, Prelude peut fournir une vision globale du niveau de sécurité du système et ainsi prévenir les attaques, intrusions et autres infections virales.



5.2 Fonctionnalités

Les différentes étapes opérées par Prelude :

Collecte Prelude collecte ces informations depuis le SI par le biais des journaux système (pare feux, routeurs, serveurs, bases de données ...) ou depuis des sondes (Snort, Suricata, Bro...). Nativement il prend en compte différents formats (syslog, Traps SNMP, fichiers plats, OPSEC, formats propriétaires, etc.) ou encore le format IDMEF (Intrusion Detection Message Exchange Format). Ce dernier est par exemple utilisable pour stocker les informations recueillies par Snort et Suricata.

Sélection Prelude utilise des critères de sélection sur les informations collectées en fonction du type, du niveau de l'alerte, de la source etc ...

Normalisation Les traces brutes sont stockées sans modification pour garder leur valeur juridique, c'est à dire utilisables en cas de litige avec les personnes responsables de l'attaque par exemple. Ces traces sont généralement copiées puis normalisées sous un format plus lisible. En effet, la normalisation permet de faire des recherches mutli-critères, sur un champ ou sur une date. Ce sont ces évènements qui seront enrichis avec d'autres données puis envoyés vers le moteur de corrélation.

Corrélation Les règles de corrélation permettent d'identifier un évènement qui a causé la génération de plusieurs alertes en amont sur le système. Cette corrélation permet de cibler plus précisément une attaque, une intrusion ou encore une attaque virale en regroupant plusieurs informations issues de différents outils du SI ou des sondes.

Une fois les informations collectées, normalisées et mis en relation, Prelude va archiver, analyser et générer des alertes.

Archivage Prelude collecte l'ensemble des journaux du système d'informations, les stocke et les indexes dans une base de données. Il offre ainsi aux opérateurs des moyens de recherche dans ces journaux lors des audits de sécurité.

Analyse Prelude propose des interfaces d'analyse avancées et graphiques sur l'ensemble des données temps réel et archivées pour assister les opérateurs dans leurs travaux d'analyse. Ce module permet de générer des rapports dans divers formats, par exemple à destination des directions de l'entreprise.

Alerte Prelude collecte, analyse et corrèle l'ensemble des informations du système et en extrait les événements suspects en temps réels pour alerter les opérateurs d'une éventuelle intrusion.

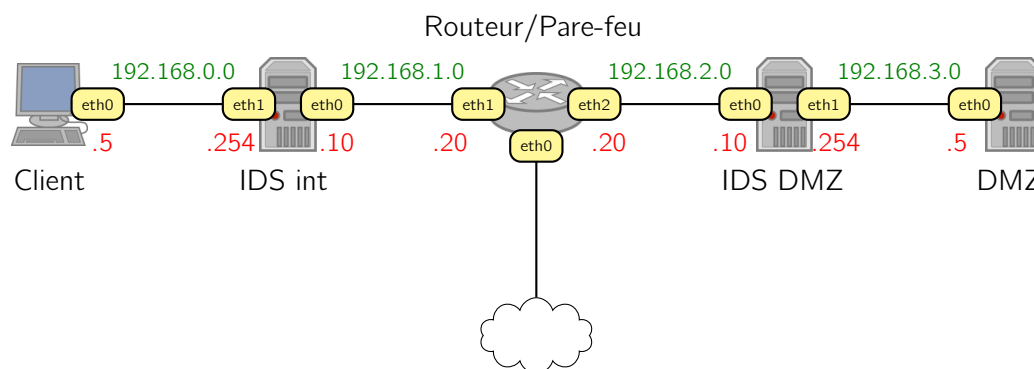
5.3 À propos

En France, la revue bimestrielle MISC est le magazine de référence en matière de sécurité informatique. Chacun des numéros possède un dossier, c'est à dire un ensemble d'articles traitant d'un sujet précis. Le dossier du numéro 69, publié le 30 Août 2013, est dédié aux SIEM et à la détection d'intrusions. Le sujet d'un article est "SIEM/IDS : L'union fait-elle la force?". Cet article tente de répondre à cette

question afin de montrer si la mise en place et la gestion d'un SIEM est réellement un avantage au sein de processus de détection d'intrusions. Cette étude a été faite en observant plusieurs sondes IDS en action lors de la détection d'une attaque web ciblée : l'exploitation d'une faille de type LFI (Local File Inclusion) identifiée par une CVE. Dans un premier temps l'étude détecte l'exploitation de cette faille au niveau réseau avec Snort, puis au niveau du système avec OSSEC et présente les avantages de chacun des ses IDS, mais aussi leurs inconvénients. Cette étude propose ensuite les possibilités de corrélation de événements au travers d'un SIEM, OSSIM, afin d'obtenir des alertes de sécurité plus précises. En conclusion, cela permet de montrer qu'un SIEM a des avantages et facilite la gestion des événements de sécurité qui surviennent sur un SI, malgré que son utilisation ne résolve pas tous les problèmes de la détection d'intrusions.

6 Démonstration de Snort

Pour illustrer ce rapport, nous avons mis en place le réseau suivant :



Nous avons réfléchi sur cinq scénarios attaquant la DMZ. Les attaques seront toutes effectuées depuis le client pour simplifier la compréhension des attaques.

Les règles se trouvent sur l'IDS DMZ dans le fichier `/etc/snort/rules/demo.rules`

6.1 Scan de ports

Premièrement, nous souhaitons détecter des scan de nmap qui utilise le protocole ICMP pour faire des scans. Pour cela, nous mettons en place la règle suivante :

```
1 alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DEMO-ATTACKS Scan NMAP"; \
2 dsize: 0; sid:7348;)
```

La seule différence avec un ping, c'est que la taille des données est vide (dsize à 0).

6.2 Injection SQL dans l'URL

Pour tester l'alerte d'injection, nous avons écrit une petite règle qui détecte les URLs contenant le motif `OR 1=1` qui est souvent utilisé dans les injections SQL de base :

```
1 alert tcp any any -> $HTTP_SERVERS $HTTP_PORTS (msg:"DEMO-ATTACKS SQL injection"; \
2 uricontent:"OR 1=1"; \sid:6969;)
```

Le champs `uricontent` est inspecté à la recherche du motif.

6.3 Injection XSS

Pour cette attaque, nous testons si le client envoie dans le contenu du paquet le motif `<script` pour détecter les injections :

```
1 alert tcp any any -> $HTTP_SERVERS $HTTP_PORTS (msg:"DEMO-ATTACKS XSS attack"; \
2 content:"<script"; sid:7373;)
```

Ici, c'est le champs `content` qui nous permet d'analyser le contenu du paquet et d'y rechercher le motif d'injection XSS.

6.4 Erreurs 403

Les erreurs 403 sont dangereuses car elles indiquent à l'attaquant que la ressource existe mais qu'il n'y a pas accès. Il est alors important de journaliser cet événement au cas où l'attaquant réussirait une attaque ultérieure. Pour cela, il faut analyser le trafic sortant du site web :

```
1 alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any ( \
2 msg:"DEMO-ATTACKS 403 Forbidden"; content:"HTTP/1.1 403"; sid:73421;)
```

On repère les erreurs 403 en analysant simplement le contenu de la page avec le champs `content`.

6.5 Fuite de /etc/passwd

On souhaite maintenant journaliser les tentatives d'accès au fichier `/etc/passwd`. Pour cela, nous analysons l'URI et le contenu du paquet :

```
1 alert tcp any any -> $HTTP_SERVERS $HTTP_PORTS (msg:"DEMO-ATTACKS /etc/passwd"; \
2 uricontent:"/etc/passwd"; content:"/etc/passwd"; sid:8455;)
```

7 Ressources

Les documents et le lab NetKit de ce projet sont disponibles sur le dépôt git : http://github.com/legrajul/projet_reseau