Les classes PSM perl

CSONE

Julien VAUBOURG

12 juin 2009

CS One - 42 avenue Montaigne 75008 Paris
Cabinet de consultants en architectures réseaux, sécurité et téléphonie IP

Table des matières

1 Préambule

1.1 Introduction

La CSonde est un outils capable de générer des rapports d'audit détaillés sur les différents flux rencontrés sur un réseau.

La capture se fait avec tcpdump (un sniffeur opensource en ligne de commande) à partir de la machine lorsqu'elle est branchée sur le port mirroring d'un switch placé à un endroit stratégique du réseau à observer. La capture dure généralement une semaine, et génére toute une série de fichier de type pcap dans le répertoire ***dumps/*** de la sonde.

Pour traiter ces fichiers pcap et arriver à un rapport PDF sur les flux réseaux rencontrés, une collection de classes perl a été développée. Pour fonctionner correctement, il faut au préalable avoir configuré le fichier ***/home/sonde/config/zones.h*** qui contient les informations liées à la composition des zones que l'on souhaite observer. Une zone est composée d'adresses IP, de sous-réseau (ex: 192.168.107.0/255.255.255.0), d'intervales d'IP (ex: 192.168.107.2-9) ou/et d'intervales de sous-réseaux (ex: 192.168.102-108.0/255.255.255.0). Le fichier ***zones.hash*** contient également la composition des GZones, qui sont des rassemblement de zones, définies par l'identifiant qui leur a été attribué (voir ***classes/PSM/Zones.pm*** pour plus d'explications sur les zones et les GZones). Ce fichier peut être configuré à partir d'une interface web disponible depuis le localhost de la CSonde.

1.2 Les répertoires de la sonde

Le répertoire de la sonde correspond à ***/home/sonde/***.

Il contient une arborescence de dossiers nécessaire au bon fonctionnement du système d'édition des rapports :

1. ***camemberts/*** : La génération des graphiques de type camemberts (pie charts) se fait à l'aide d'une petite application java. Ce dossier contient cette application (Camembert), la source de l'application et les librairies nécessaire (JFreeChart et PNGEncoder).
2. ***classes/*** : C'est le répertoire qui devra être ajouté au PATH de perl dans les entêtes des scripts. Il contient le dossier ***PSM/*** qui contient toutes les classes perl. Pour une explication detaillées du fonctionnement de chacun de ces classes, se référer aux entêtes de leurs fichiers. Nous verrons les interactions entre elles dans la suite de cette documentation.
3. ***config/*** : Répertoire des fichiers de configuration générés par l'interface web. Nous reviendront sur leur syntaxe.
4. ***dumps/*** : C'est le répertoire dans lequel écrit tcpdump pendant la période de capture des données sur le réseau. Il contient les fichiers au format pcap qui seront traités par la classe ***Capture.pm***, qui ira extraire les entêtes des paquets pour les stocker dans un dossier ***dumps/text/*** avec les mêmes nom de fichier que les dumps correspondants ormis des zéros rajoutés pour rectifier l'ordre lexicographique.
5. ***rapport/*** : Il contient le fichier ***rapport.tpl*** qui sert de base au document LaTeX qui sera généré dans ce même répertoire (en .tex). Le rapport final en .pdf sera aussi généré dans ce même répertoire.
6. ***rrd/*** : Lors du traitement des fichiers pcap, après avoir généré les fichiers d'entêtes dans ***dumps/text/***, ***Capture.pm*** a pour charge de tirer de ces fichiers textes une arborescence de fichiers RRD (Round Robin Data), qui seront créés dans ce dossier. Cette arborescence contient des dossiers d'adresses IP au premier niveau qui sont des IP sources, puis dans chacun de ces dossiers, un second niveau de dossiers qui ont pour nom les IP destination, et dans ceux-ci des fichiers RRD des flux dans le temps de la source vers la destination. Les fichiers RRD indiquent dans leur nom le protocole (***PROTO_*.rrd***) ou le port (***PORTS*.rrd***) qu'ils ont observé en particulier. Sur le premier niveau de l'arborescence on trouve aussi un fichier ***PROTO_*** et un fichier ***PORTS*** qui résument tous les protocoles et ports trouvés pour l'ensemble des flux. Ce sont les groupes de possibilités (voir la fonction requete() de ***Zones.pm***).
7. ***scripts/*** : Enfin, pour faire fonctionner tout ça il faut des scripts, et c'est dans ce dossier que nous les trouverons. Il seront détaillés dans la suite de cette documentation.

2 Mécanisme complet

2.1 La collecte des données

2.1.1 Hardware

La première étape pour l'audit réseau est de capturer les données qui transitent sur le réseau.

La CSOnde est une machine de type PSMBox, équipée d'un processeur P4, de 512MB de RAM, d'un disque dur de 500Go et d'une carte réseau GigaBit. Pour collecter les données en entreprise, elle se branche sur le port mirroring d'un switch qu'on intercale à une place stratégique du réseau. Ainsi, la fiabilité du switch assure que le réseau ne peut pas être perturbé par notre audit et le mirroring permet de récolter l'ensemble des données qui passent par le switch, en toute transparence. La configuration de vlan sur le switch permet d'auditer plusieurs sous-réseaux en même temps. L'avantage de prendre notre propre switch est d'être assuré d'avoir un équipement qui supporte le mirroring, il peut être facultatif si le client utilise un switch bien placé qui permet cette fonctionnalité. La CSonde est configurée pour vérifier toutes les cinq minutes que la capture est toujours en cours (tâche CRON) et redémarrera automatiquement en cas de coupure de courant (configuration du BIOS). La CSonde utilise un système Debian, utilisé avec les paquets de base (5.0 Lenny, au moment d'écrire cette doc).

2.1.2 Un sniffeur : tcpdump

Les données sont collectées à l'aide de tcpdump (un sniffeur de réseaux), qui est lancé au démarrage de la machine par un service qui s'appelle ***sonde*** et qui se trouve dans ***/etc/init.d/***. Tcpdump est configuré pour créer des fichiers prefixés par ***dump*** suivi du timestamp unix (epoch) de l'instant où il le crée, suivi d'un underscore et enfin un nombre. Ces fichiers sont des fichiers binaires de type pcap qui contiennent l'intégralité des paquets (capturés par la librairie pcap) et sont stockés dans le dossier ***/home/sonde/dumps/***. Chaque fois que le fichier de dump en cours atteint la taille maximal d'un giga-octets, tcpdump en démarre un nouveau du même nom en incrémentant le nombre à la fin. Le premier fichier finit simplement par un underscore au lieu d'un zéro.

Tcpdump a été choisi pour sa simplicité d'utilisation, et son large choix d'options. En collectant l'ensemble des information des paquets, il permet un traitement des données en toute liberté par la suite sans risque de devoir retourner chez le client faire une semaine de capture. Il propose également un système de filtres puissant qui permet de n'extraire que les paquets qui nous intéressent (tcpdump est appellé dans ***Capture.pm***).

Pour plus de détails concernant la configuration de la CSonde, se référer à la doc d'installation.

2.2 Le traitement des dumps

2.2.1 Construction de l'arborescence des fichiers RRD

Une fois que la CSonde revient à la maison mére, il faut traiter les fichiers de capture de tcpdump pour les transformer en une arborescence de bases RRD.

Avant d'arriver aux fichiers RRD, il faut extraire les paquets des binaires et ne stocker que leurs entêtes dans des fichiers textes. Cette mission revient aux fonctions de la classe ***Capture.pm***.

Le script ***capture.pl*** du dossier ***scripts/*** utilise cette classe et accepte trois options :

1. `./capture.pl bin2txt` : Transforme les fichiers binaires du répertoire ***dumps/*** en fichier textes d'entêtes de paquets dans le dossier ***dumps/text/***. C'est tcpdump qui s'occupe de cette lecture des binaire, notamment avec l'option `-r`.
2. `./capture.pl rrd` : Transforme les fichiers d'entêtes de paquets de ***dumps/text/*** en une arborescence de bases RRD (voir la description du dossier ***rrd/*** dans l'introduction).
3. `./capture.pl all` : Transforme les binaires en fichiers textes d'entêtes et construit les bases RRD (`bin2txt` puis `rrd`)

Le fonctionnement de ***capture.pl*** est imagé dans ***capture.png***.

2.2.2 Les fichiers RRD

Les fichiers RRD sont des bases de données temporelles. Elles ne contiennent qu'une suite de rapport temps:valeur. Chaque base RRD est un fichier construit à l'aide l'option `create` de rrdtool, un outils de traitement de ce format de bases de données. Chaque RRD contient des sources de données (DS) et des archives RRA. La source de donnée prend des valeurs en fonction du temps et les archives RRA permettent de les traiter de la façon pour laquelle elles sont prévues. Nous utilisons les archives RRA en mode AVERAGE : chaque valeur renseignée dans le RRD (via `rrdtool update`) doit être supérieures à la dernière valeur rentrée (en fonction du temps indiqué pour la valeur). L'archive RRA se charge de faire la différence des valeurs dans le temps afin de pouvoir donner la possibilité de visualiser l'évolution de la donnée au cours du temps (notamment au travers des graphiques, tracés avec `rrdtool graph`). Nos RRD ne contiennent qu'une source DS chacun : elle se nomme `cpt` et représente la donnée indiquée dans le nom du fichier (ex: ***PROTO_UDP.rrd*** pour les flux qui ont transité avec le protocole UDP de la source vers la destination des répertoires dans lesquels il est situé). Un fichier RRD a une taille fixe, selon le nombre d'entrées à

garder indiqué à sa création, il fait des moyennes des valeurs les plus anciennes, qui deviennent donc de moins en moins précises.

Créer une base RRD :

```
rrdtool create rrd/ipSrc/ipDest/PROTO_UDP.rrd
-step 5 -start 123456789 -end 223456789
DS:cpt:COUNTER:1200:U:U
RRA:AVERAGE:0.5:1:600
RRA:MAX:0.5:1:600
```

Le step correspond à la précision qu'on gardera lorsqu'on lira la base (avec un step 5, la base nous renverra une valeur pour toutes les 5 secondes). start et end correspondent à la période d'enregistrement des RRD (typiquement on mettra les timestamp de début et de fin d'analyse de la CSonde sur le réseau du client). Le 1200 correspond au temps maximum en secondes entre chaque valeur entrée. Si deux valeurs sont entrées avec plus de 1200 secondes de différence (en fonction de leur timestamp), on considérera la valeur nulle (UKN). Le 600 de AVERAGE et MAX, lui, indique le nombre de fois qu'on gardera 1200 secondes de données, soit ((600*1200)/1600)/24 = 8, soit une semaine et un jour. Au delà, les premieres données sont transformées petit à petit en moyennes et deviennent de plus en plus imprécises.

Mettre à jour :

```
rrdtool update rrd/ipSrc/ipDest/PROTO_UDP.rrd
timestamp:valeur timestamp:valeur timestamp:valeur
```

Timestamp est un timstamp unix (epoch) et la valeur doit toujours être croissante, ici on rentre trois valeurs d'un coup.

Tracer un graphique :

```
rrdtool graph /tmp/ns-graph-123456789.png
DEF:c1=rrd/ipSrc/ipDest/PROTO_UDP.rrd:cpt:AVERAGE
CDEF:c1EnOctets=c1,8,/
AREA:c1EnOctets#FF0000:"Le TCP de ipSrc vers ipDest"
```

Explications :

1. DEF : Une courbe c1 est définie, elle prendra pour valeurs celles que nous renverra la RRA qui gére les moyennes (AVERAGE) pour la DS cpt qui a été définie à la création de la base.
2. CDEF : Les données sont en bits, on les souhaite en octets. On crée donc une CDEF qui effectuera un calcul à partir d'une expression RPN. La syntaxe RPN se lit ainsi : prendre c1 prendre 8 et diviser l'un par l'autre. Cette CDEF devient une courbe à part entière qui s'appellera c1EnOctets.
3. AREA : On souhaite maintenant afficher c1EnOctets sur le graphique. Avec AREA, on lui dit qu'on veut un graphique plein au lieu de lignes (LINE suivi d'une taille pour tracer des lignes), et on indique la couleur en hexadécimal (ici rouge). Enfin vient le label (facultatif) qui apparaitera avec un petit carré de la même couleur sous le graphique.

On peut bien entendu afficher autant de courbes qu'on le souhaite sur un graphique. Pour additionner deux courbes (donc deux RRD), il faut faire une CDEF qui additionne deux DEF (c1,c2,+) et l'afficher. Lorsque ***Graphique.pm*** trace un graphique à partir d'un objet GroupeRRD, elle additionne les sources du groupe de cette façon pour afficher une seule courbe (ex: addition des protocoles TCP et UDP pour avoir la bande passante).

2.3 La génération du rapport

2.3.1 Création d'un format TEX

1. `./rapport.pl poissy.tex` : Le rapport final, généré dans rapport/, s'appellera poissy.tex et le PDF poissy.pdf. Cet argument doit finir par .tex, est facultatif (par defaut, rapport.tex) et doit toujours se trouver en premier.
2. `./rapport.pl nocache` : Lorsque le cache est activé (par défaut), les instances de PSM::RRD sont archivées et partagées lorsqu'on souhaite en recréer une qui a déjà été générée. Les requêtes de la fonction requete() de PSM::Zones sont également archivées pour répondre plus vite lorsqu'on souhaite la même formation de RRD/GroupeRRD. Enfin, les images des graphiques et camemberts sont récupérées si elles ont déjà été tracées. L'option nocache désactive ces trois fonctionnalités.

3. `./rapport.pl debug` : Le débugage est désactivé par défaut. L'option debug permet d'afficher sur la sortie standard les warnings et la gestion du cache (mise en cache ou récuperation du cache). Les warnings interviennent surtout lorsqu'on crée un PSM::RRD ou PSM::GroupeRRD qui n'a aucune source d'existante. Les warnings sont naturels lorsqu'on fait des requêtes par type, il ne faut pas chercher à les éliminer, ils permettent juste de voir ce qui se passe.

Toutes ces options peuvent se combiner, et ormis le nom du fichier tex qui doit toujours être en premier (mais qui est facultatif), les autres options peuvent être définies dans n'importe quel ordre.

2.3.2 Vers le format PDF

1. `./requete.pl pdf` : Le fichier .tex sera automatiquement compilé en PDF dans le répertoire rapport/. Deux compilations successives sont effectuées (pour la table des matiêres et le nombre de page, c'est un comportement normal de LaTeX).

Cette dernière option transforme le fichier TEX en PDF immédiatement après sa création.

Le fonctionnement de ***rapport.pl*** est imagé dans ***rapport.png***.

3 Les fichiers de config/

3.1 zones.hash

Ce fichier de configuration est généré par l'interface web lors de la définition des zones et GZones et est utilisé dans la classe ***Capture.pm***. Il est mis en mémoire (évalué) par la fonction `lire()` de cette classe.

Une zone est un ensemble d'adresses IP et de sous-réseaux (definis avec leurs masques, en octal : 255.255.255.0) qui seront traités sous un seul nom. Chaque zone a une référence unique (entier). Une IP peut être dans plusieurs zones (ex: 192.168.107.2 dans une zone et 192.168.107.0/255.255.255.0 dans une autre).

Les zones sont définies dans une hash map écrite en dure dans le fichier :

```
%Zones::zones = (
'refZoneInt0' => ['Nom0', ['ip01', 'ip02', 'subnet01/netmask01', 'ip03']],
'refZoneInt1' => ['Nom1', ['subnet11/netmask11', 'ip11']]
);
```

Une GZone est un groupe de références de zones qui seront traitées sous un seul nom. Chaque GZone a une référence unique (entier).

Les GZones sont définies elles aussi en dure dans le même fichier :

```
%Zones::GZones = (
'refGZoneInt0' => ['GNom0', [refZoneInt0, refZoneInt1, refZoneInt2]],
'refGZoneInt1' => ['GNom1', [refZoneInt0, refZoneInt1]]
);
```

La GZone reseau est crée dynamiquement et contient les références de toutes les zones. Cette GZone represente tout ce qui n'est pas internet. Tout ce qui n'est pas dans réseau est donc internet.

4 Documentation images

La documentation propose également quatre PNG :

1. ***classes.png*** : Diagramme de classes UML des classes de PSM
2. ***requete.png*** : Explication séquentielle de la fonction requete() de ***Zones.pm***
3. ***capture.png*** : Explication séquentielle du fonctionnement du script ***capture.pl***
4. ***rapport.png*** : Explication séquentielle du fonctionnement du script ***rapport.pl***

Ces images ont été créées avec Bouml 4.4.2. Pour les modifier, ouvrez le fichier ***imagesdoc.prj*** du répertoire des sources de la doc, choissisez le diagramme à gauche, et cliquer à droite dans le blanc du diagramme pour choisir "Save optimal picture part (PNG)", après avoir effectué les modifications.