Lsfw, outil de test de règles de pare-feux distribués sur un réseau

Patrick Lamaizière
Centre de Ressources Informatiques, Université de Rennes 1
Bat 12D Campus de Beaulieu – Avenue du Général Leclerc - 35042 Rennes Cedex

Résumé

Les pare-feux sont des composants de sécurité importants dans les architectures réseaux. De même que la mauvaise configuration d'un routeur peut produire des problèmes imprévisibles de routage, des pare-feux mal configurés peuvent échouer à assurer la politique de sécurité attendue.

Ces pare-feux sont distribués en plusieurs points du réseau et coopèrent pour assurer une politique de sécurité globale. Comme les règles de filtrages de ces pare-feux se cumulent entre deux points d'un réseau, il devient difficile de prédire le fonctionnement qui en résulte de bout en bout, et s'il correspond à la politique de sécurité de plus haut niveau.

Lsfw (pour list firewall) est une application développée et utilisée au CRI de l'Université de Rennes 1 pour aider les administrateurs réseaux dans cette problématique. L'application interprète les fichiers de configuration natifs des équipements réseaux et émule leur comportement au niveau IP (filtrage et routage). Trois types d'équipements sont implémentés : Packet-Filter, Cisco IOS et Cisco PIX.

Mots clefs

Sécurité, Pare-feux

1 Introduction

Au niveau du réseau, l'Université de Rennes 1 (UR1) ce sont :

- 273 sous-réseaux ;
- 15 routeurs ;
- ~20 000 lignes de configuration.

Le CRI disposait auparavant d'un outil similaire à Lsfw, déjà développé en interne. Mais cet outil était très spécifique au réseau de l'UR1 et ne traitait que des configurations Cisco.

Dans le cadre de la migration de matériel sous Packet Filter / OpenBSD, il était nécessaire de développer un nouvel outil, idéalement plus propre, modulaire, extensible et réutilisable par d'autres.

Lsfw a été développé de mars à août 2010 lors de mon stage de fin d'études en master Sécurité des Systèmes d'Informations (SSI).

Cet article présente l'outil d'une manière pratique en se basant sur un exemple d'architecture réseau. Nous verrons comment s'utilise l'outil, ce qu'on peut en attendre et comment il se configure.

Pour finir je présenterai brièvement les principes et l'implémentation de l'outil.

2 Lsfw

Pour illustrer l'application, nous nous baserons sur le réseau suivant. Il se compose :

- d'un routeur Cisco IOS (R1) ;
- d'un pare-feu Packet Filter sous OpenBSD (P1).

P1 isole le réseau interne (192.168.1.0/24) et héberge une DMZ (192.168.0.0/24) avec deux serveurs (web et mail). La politique de sécurité est simple -et à ne pas prendre en exemple- :

- ICMP permis partout ;
- trafic autorisé entre le réseau interne et la DMZ, seulement sur les ports nécessaires (HTTP, SMTP, IMAP…).

Les fichiers de configuration sont disponibles sur https://subversion.cru.fr/jtacl/trunk/jtacl/doc/tutojres/

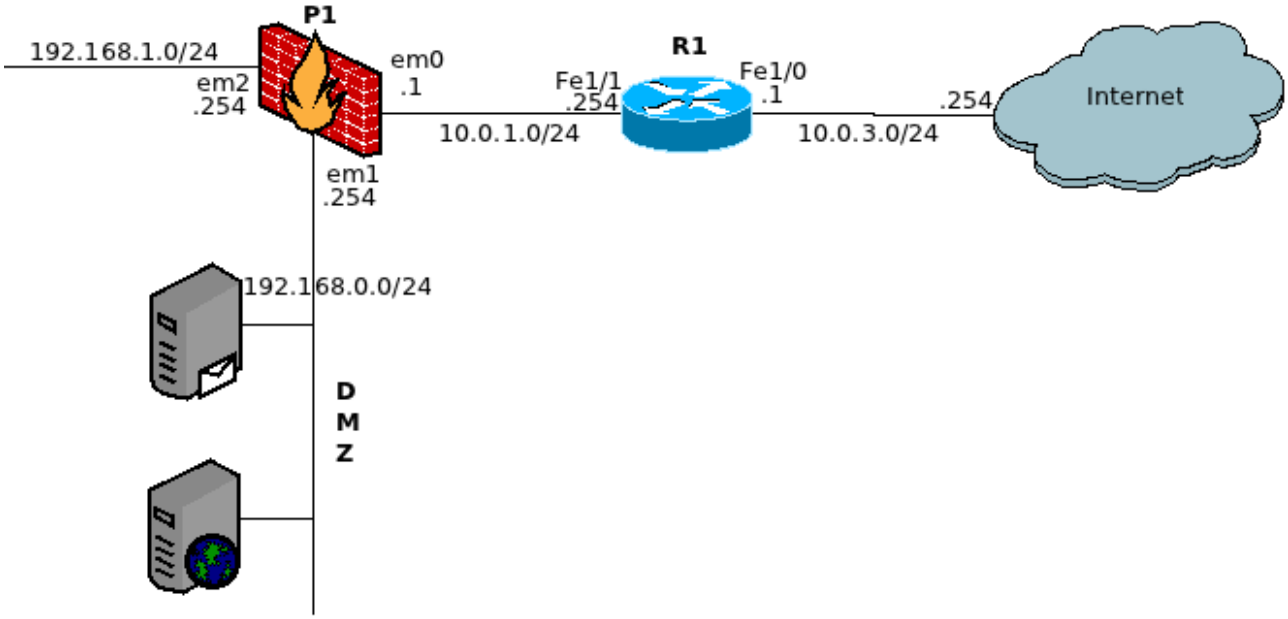

Figure 1: Réseau d'illustration

2.1 Utilisation

Lsfw est un outil texte et utilise un shell interne pour interagir avec l'utilisateur.

Probe

La commande la plus utile est la commande « probe » qui permet de sonder les règles de filtrages des équipements. La commande prend au moins une adresse IP source et une adresse IP destination et affiche les règles qui correspondent à la commande entre ces deux points du réseau. L'application utilise des « sondes » pour interroger les équipements, une sonde est l'équivalent d'un paquet IP mais en diffère sensiblement.

Par exemple, pour lister les règles qui s'appliquent pour un hôte sur le réseau interne (192.168.1.1) et internet (une IP quelconque) cela donne simplement :

```
lsfw> probe 192.168.1.1 1.2.3.4
```

L'application recherche un réseau qui contient l'adresse source et injecte une sonde sur l'équipement connecté à ce réseau (ici P1, interface em0). Lsfw utilise des adresses IP (v4 ou v6) exprimées en notation CIDR (i.e. préfixe/longueur) et ne fait pas de différence entre un hôte et un réseau. Un hôte est simplement un réseau de taille un. Ce qui nous permet d'exprimer des requêtes avec des réseaux, par exemple :

```
lsfw> probe 192.168.1.0/24 1.2.3.4
```

S'il y a ambiguïté sur l'équipement source, il est nécessaire de le spécifier. Ceci arrive sur les réseaux d'interconnexion :

```
lsfw> probe 10.0.1.1 1.2.3.4
Too many links
```

Dans le cas ci-dessus, on ne sait pas quel équipement utiliser puisque le sous-réseau contenant 10.0.1.1 est connecté à P1 et R1. Lsfw émet l'erreur « too many links (trop de liens) ». Il est nécessaire de spécifier l'équipement d'interconnexion qu'il s'agit de traverser :

```
lsfw> probe on R1 10.0.1.1 1.2.3.4
```

Il est aussi parfois souhaitable de spécifier un équipement à partir duquel doit commencer le sondage, par exemple pour faire des tests de règles d'anti-spoofing, ou pour lister toutes les règles quelle que soit l'IP source. L'exemple suivant utilise une sonde d'IP source « tout IPv4 (0/0)», et qui est injectée sur P1, via l'interface em2.

```
lsfw> probe on P1|em2 0/0 1.2.3.4
```

La commande probe possède d'autres options pour préciser la requête (protocole, ports, type ICMP, drapeaux TCP) sachant que plus la requête sera précise, plus la réponse le sera également. La commande probe suivante effectue un sondage pour du TCP, depuis un port dynamique (> 32767) vers le port HTTP, avec le drapeau TCP SYN à 1 (début de connexion TCP) :

```
lsfw> probe 192.168.1.0/24 192.168.0.3/24 tcp dyn:http flags S
```

Résultat

Le résultat d'une commande probe comprend trois parties. La première partie décrit le chemin suivi par la sonde. Ensuite sont affichées les règles qui correspondent à la requête. Enfin un résumé indiquant le résultat global du test, c'est à dire si le trafic passe ou pas.

Prenons l'exemple d'une requête vers le serveur web depuis internet. Lsfw affiche le chemin suivi équipement par équipement et interface par interface :

```
lsfw> probe on R1|10.0.3.1 1.2.3.4 192.168.0.2 tcp dyn:http
################################
-------- Routed probes --------
Path:
On: R1 (cisco router #1)
   FastEthernet1/0 (internet)
      interface IP: 10.0.3.1 network: 10.0.3.0/24
   FastEthernet1/1 (INTERNAL)
      interface IP: 10.0.1.254 network: 10.0.1.0/24
      nexthop: 10.0.1.1
On: P1 (firewall #1)
   em0 (em0)
      interface IP: 10.0.1.1 network: 10.0.1.0/24
   em1 (em1)
      interface IP: 192.168.0.254 network: 192.168.0.0/24
      nexthop: 192.168.0.0/24
```

Ensuite, les règles correspondant à la requête, équipement par équipement. Chaque règle est affichée sous la forme « action (ACCEPT/DENY)» « fichier (le fichier où est définie la règle » « texte (la règle)» :

```
------
R1 (cisco router #1)
Matching ACL on input: FastEthernet1/0 (internet)
 ACCEPT  r1-conf #34: [INTERNET_IN] permit ip any any
```

```
  DENY  [INTERNET_IN] *** implicit deny ***
Matching ACL on output: FastEthernet1/1 (INTERNAL)
 ACCEPT  r1-conf #69: [INTERNAL_OUT] permit ip any any
  DENY  [INTERNAL_OUT] *** implicit deny ***
------
P1 (firewall #1)
Matching ACL on input: em0 (em0)
  DENY  ./conf/pf.conf #26: block all
 ACCEPT  ./conf/pf.conf #33: pass proto tcp from any to $httpserver port { http, https }
Matching ACL on output: em1 (em1)
  DENY  ./conf/pf.conf #26: block all
 ACCEPT  ./conf/pf.conf #33: pass proto tcp from any to $httpserver port { http, https }
----------------
```

Et enfin le résumé global qui indique le résultat du filtrage (Global ACL result) sur l'intégralité du chemin et le résultat du routage (Global routing result), c'est à dire si la sonde parvient jusqu'à sa destination. Le résultat n'est pas forcément bien déterminé si la requête n'est pas précise :

```
Global ACL result is: ACCEPT
Global routing result is: ROUTED
```

2.2 Configuration

Lsfw est écrit en Java 1.6 et est distribué sous la forme d'une archive Jar qui contient tout ce qui est nécessaire. L'application utilise un fichier de configuration général où l'on déclare au moins les équipements utilisés. Ensuite chaque équipement dispose de son propre fichier de configuration.

Configuration générale

Elle s'effectue par un fichier au format XML et permet de spécifier :

- les équipements utilisés ;
- la topologie du réseau ;
- quelques options globales.

Un équipement est déclaré par une entité XML « equipment » :

- le type de l'équipement (c'est à dire la classe Java qui l'implémente) ;
- un nom et un commentaire ;
- un fichier de configuration.

Par exemple le routeur R1 est déclaré par l'entité :

```
<!-- Cisco IOS Router -->
<equipment
   classname="fr.univrennes1.cri.jtacl.equipments.cisco.router.CiscoRouter"
   name="R1"
   comment="cisco router #1"
   filename="./r1.xml"
```

```
/>
```

Topologie

Sur un vrai réseau, un équipement n'a aucune connaissance de la topologie (c'est à dire qui est connecté à qui). Lsfw détermine la topologie automatiquement en considérant que des équipements qui possèdent un même sous-réseau sont alors connectés entre eux. Ceci n'est pas forcément vrai, par exemple pour des réseaux privés. En conséquence il peut être nécessaire de préciser des liens de topologie manuellement.

Une autre nécessité est de spécifier les « bordures » du réseau. Quand une sonde arrive sur une bordure, on considère qu'elle est arrivée à destination. Dans notre réseau exemple, un lien de bordure est spécifié sur le sous-réseau coté internet du routeur R1 pour indiquer que notre réseau s'arrête ici. Ceci se fait par l'intermédiaire d'un « lien topologique » :

```
<tlink network="10.0.3.0/24" topology="R1|10.0.3.1" border="true" />
```

Ce lien topologique exprime que le réseau 10.0.3.0/24, sur la patte du routeur R1 d'IP 10.0.3.1, est une bordure et que le trafic s'arrête donc là.

Équipement

Chaque équipement est configuré via un fichier XML. Ce fichier indique le fichier de configuration natif de l'équipement et d'autres options spécifiques.

Par exemple le pare-feu P1 est configuré de la sorte :

```
<equipment>
  <!-- interfaces -->
  <iface name="em0" comment="em0" ip="10.0.1.1" network="10.0.1.0/24" />
  <iface name="em1" comment="em1" ip="192.168.0.254" network="192.168.0.0/24" />
  <iface name="em2" comment="em2" ip="192.168.1.254" network="192.168.1.0/24" />

  <!-- PacketFilter file (pf.conf) -->
  <pfconf filename="./conf/pf.conf" />

 <!-- routing engine -->
  <routing>
    <route prefix="0.0.0.0/0" nexthop="10.0.1.254" />
  </routing>
</equipment>
```

On y trouve la définition des interfaces, le chemin vers le fichier de règles (pf.conf) et l'ajout d'une route par défaut vers le routeur R1.

C'est encore plus simple pour le matériel Cisco puisqu'il suffit de spécifier le fichier de configuration natif, les interfaces et les routes en sont extraites.

2.3 Implémentation

L'application émule le comportement des équipements face à un pseudo paquet IP (des sondes), et les fait fonctionner sur un modèle de réseau IP. Le modèle du réseau est simple et, sans surprise, calqué sur un réseau réel. Il est constitué par des équipements reliés par des liens. Le cœur de l'application joue le rôle de la couche physique en transmettant les paquets d'équipement en équipement, en fonction de la topologie.

Un équipement, quant à lui, est connecté à ce réseau et doit être capable :

- d'accepter un paquet ;

- de le filtrer ;
- de le router ;
- et enfin de le renvoyer sur le réseau.

Ainsi, il y a très peu de pré-requis sur l'implémentation de l'équipement. La seule nécessité est qu'il sache communiquer sur le réseau modélisé. Pour homogénéiser l'application tant au niveau de l'implémentation que de la documentation, un équipement type est proposé. Cet équipement inclut divers services utiles (moteur de routage, base de données de noms pour les services ou l'ICMP, etc.) et l'implémentation d'un équipement n'a plus qu'à s'occuper de l'interprétation des fichiers de configurations et de l'émulation du comportement.

La difficulté de la programmation d'un équipement dépend beaucoup de la complexité de ce dernier. Pour avoir un ordre d'idée, l'implémentation de Packet Filter (qui est riche en fonctionnalités) fait environ 11 000 lignes auxquels s'ajoutent 3 500 lignes pour les tests unitaires (test de la grammaire) et le temps de développement nécessaire a été d'un mois. Le routeur IOS Cisco ne fait que 3 700 lignes et 400 lignes de test.

Une branche (jtacl-pfconv) de l'application a été développée spécifiquement pour nous aider à migrer des règles de matériels Cisco vers Packet Filter. Cette version, associée à des suites de tests pour comparer le résultat par rapport aux règles d'origines, nous a été très utile pour effectuer la migration en minimisant les erreurs possibles.

3 Conclusion

Un des buts de l'outil était de fournir un outil évolutif, libre et documenté. L'outil et la documentation sont disponibles en ligne sur la forge de Renater https://listes.cru.fr/wiki/jtacl/. L'outil est en double licence Esup-Portail / BSD.

Lsfw n'est pas parfait (voir les limitations dans la documentation) et on ne peut garantir qu'il fonctionnera sur n'importe quel réseau du fait des nombreuses variantes dans les configurations des matériels. Il est toutefois utilisé quotidiennement à l'UR1 avec satisfaction, ainsi qu'à L'IRISA (Rennes).