TP Sécurité

Par Landry Breuil (breuil at craig dot fr) *Sujet de TP sous*  *licence CC-BY-NC-SA*

Préambule

Vous travaillez par binôme sur un poste, et en collaboration avec les binômes voisins - c'est à vous de vous répartir les tâches (un groupe fait la partie serveur, l'autre la partie client puis vice-versa, un commence par la doc de telle techno, l'autre groupe attaque l'autre techno pendant ce temps, etc...)

Il est attendu un rapport sur l'ensemble du TP par ilôt, et rendu au plus tard le lundi en 15 après le dernier jour de TP. Les rapports sont faits pour s'assurer de la compréhension des problématiques abordées dans le TP, et de votre capacité de synthèse - pensez à répondre aux questions présentes dans l'énoncé...

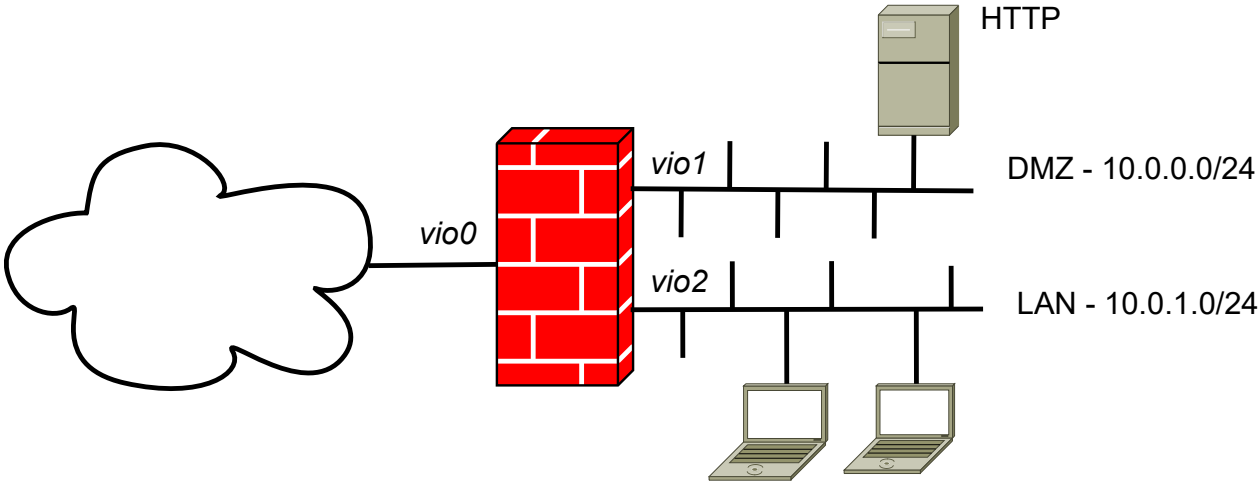
En 4 tp, nous allons reproduire (en partie) un schéma réseau classique d'une PME, le tout en utilisant des machines virtuelles sous VirtualBox.

Elles seront (au début) au nombre de 4:

- Une machine OpenBSD, qui fera office de firewall/coeur de réseau/serveur DHCP/serveur VPN. C'est cette machine sur laquelle la plupart des configurations auront lieu. Par souci pratique, habituez vous a vous y connecter en SSH.. et utilisez tmux(1) pour multiplexer les connections.
- Une machine Debian qui sera dans une zone dite 'DMZ' et servira de serveur HTTP
- Deux machines Windows, simulant respectivement un poste de travail distant (télétravail, commercial en déplacement..) et un autre sur le réseau interne de l'entreprise.

Réseau

Chacune des machines de la salle étant dans la même plage d'adresse, on considérera que la salle correspond à un mini-Internet ou chaque poste physique voit chacun des autres postes physique, et correspond à un site distinct.



- Les postes de travail Windows auront une seule interface avec une IP publique récupérée par DHCP
- Le firewall OpenBSD aura 3 interfaces pour commencer:
  - `vio0` avec une IP en DHCP publique. C'est cette patte qui donnera accès à internet au reste du réseau.
  - `vio1` avec une IP privée statique pour la DMZ. Le serveur Debian fournissant le service HTTP sera dans cette zone.
  - `vio2` avec une IP statique pour le LAN des postes de travail internes au réseau de la PME. Au départ, les postes Windows seront dans cette zone.
- Le serveur Debian aura une seule interface en IP statique.

Installation

Sur les postes de travail, nous allons stocker les VM dans /VM. Modifier la configuration de VirtualBox pour qu'il y stocke les VM par défaut.

Importer les 4 machines virtuelles depuis celles disponibles dans `/VM/OVA/tp_archi_reseau` (ou télécharger les images sur ce drive) :

- OpenBSD 7.4 (Machine virtuelle avec 512Mo de RAM, 3Go de disque.)

  Vérifier la configuration de ses 3 cartes réseau dans la configuration de VirtualBox: la première en mode *pont* sur `eth0` , la 2e en mode *réseau interne* sur `dmz` , et la 3e en mode *réseau interne* sur `lan` . Une fois la machine démarrée, noter les adresses IP et MAC pour s'y retrouver. Attention, il faut **absolument** regénérer/réinitialiser l'adresse MAC pour l'interface publique de votre VM, sinon il pourrait y'avoir des conflits avec les autres VM de la salle.

> Par quel fichier de configuration sont configurées les addresses ces 3 interfaces ? Qu'est ce qui détermine la route par défaut utilisée par les paquets sortants de la

> machine ?

Note: le mot de passe root est *root*. `ssh` est actif par défaut, il est conseillé de l'utiliser pour se connecter a la VM, depuis un terminal sur votre poste de travail.

- Debian Bookworm 12.2 (Machine virtuelle avec 256Mo de RAM, 1.5Go de disque.)

  Par quel fichier de configuration est configurée l'addresse de son interface ?

  Note: le mot de passe root est aussi *root*. Il y'a un 2e utilisateur test avec pour mot de passe *test*. Par défaut la configuration SSH n'autorise pas root a se connecter.

- Windows 10

  (mdp admin: *student*)

- Windows 7

  (mdp admin: *gonfle!*)

Vérifier que les machines se 'voient' mutuellement, noter leurs addresses IP respectives.

Partie I : Le firewall

Il existe deux grandes familles de firewall libres: PacketFilter(PF) que l'on retrouve sur les systèmes BSD, et Netfilter/iptables que l'on retrouve sur les systèmes Linux. Dans le cadre de ce TP nous allons utiliser PF. Dans PF par défaut, **last matching rule wins**. On peut utiliser le mot-clé `quick` pour arréter la correspondance d'un paquet avec le reste des règles de filtrage, mais c'est hors du cadre de ce TP.

Sécurisation par défaut en entrant depuis internet

En étudiant la documentation de PF (faq, pfctl(8), pf.conf(5)) bloquer le trafic entrant par défaut sur `vio0`, mais laisser passer le trafic ICMP.

Ouvrir uniquement les ports nécessaires

On veut laisser accessible SSH sur le firewall, mais uniquement depuis la machine hôte (les autres postes de travail de la salle ne doivent pas pouvoir entrer), et logguer toutes les connexions que l'on a bloqué.

pflogd(8) est activé par défaut, utiliser tcpdump(8) sur l'interface réseau virtuelle `pflog0` du firewall pour vérifier que les connexions venant d'autres postes sont bien bloquées. Vous

pouvez aussi utiliser systat(1) `rules` pour avoir une idée du nombre de paquets correspondant a chaque regle PF.

NATer la DMZ

Dans Virtualbox, reconfigurer l'interface de la machine virtuelle Debian pour qu'elle soit en mode *réseau interne* sur `dmz`. Dans la VM, lui configurer une IP statique (cf interfaces(5), et ip(8)) en `10.0.0.2/24` utilisant `10.0.0.1` comme route par défaut. Vérifier qu'elle est accessible depuis le firewall OpenBSD, et vice-versa. A ce stade, la machine ne peut pas accéder à internet.

> Pouvez vous vous connecter en tant que root via ssh depuis le firewall vers le serveur debian ? Pourquoi ?

Sur le firewall OpenBSD, configurer la NAT sur l'interface `vio1` (cf pf.conf(5) et sysctl.conf(5)). Vérifier que la machine Debian peut maintenant accéder à internet. Elle reste inaccessible de l'extérieur.

> Que faut t'il configurer pour que cette machine puisse résoudre des noms DNS ?

Configurer le serveur DHCP

Sur le firewall OpenBSD, configurer dhcpd(8) sur l'interface `vio2` afin que les clients Windows aient une ip sur la plage `10.0.1.2 -> 10.0.1.10`, récupèrent le DNS public (le même que celui de la machine hôte) et utilisent le firewall comme route par défaut. Des fichiers de configuration d'exemples se trouvent dans `/etc/examples`.

Note: sur OpenBSD, les services/demons sont gérés par la commande rcctl(8).

NATer les postes de travail

Dans Virtualbox, reconfigurer l'interface des machines virtuelles Windows pour qu'elle soit en mode *réseau interne* sur `lan`. Vérifier qu'elles récupèrent bien une adresse via DHCP depuis le firewall.

Sur le firewall OpenBSD, configurer la NAT sur l'interface `vio2`. Vérifier que les postes de travail Windows connectés au LAN ont accès à internet.

Port forwarding

Configurer un port forwarding pour que la machine hôte (et uniquement elle) puisse directement accéder à la machine virtuelle Debian sur son port SSH en utilisant le port 2222. Tester que ce port n'est pas accessible depuis une autre machine de la salle.

Redirection de ports pour l'accès public

En modifiant la configuration de PF sur le firewall OpenBSD, rediriger les connexions de l'extérieur à destination du port HTTP vers le serveur debian.

Depuis l'extérieur (ie un poste 'hôte'), essayer d'accéder à l'IP publique de votre sous-réseau via un navigateur.

> Où sont stockés les fichiers journaux du serveur HTTP ? Vérifier que les connexions faites sont bien enregistrées. Si ce n'est pas le cas, que faut il faire pour logguer chaque requête faite par un client ?

Filtrage sortant

En fonction des cas d'utilisation et de la confiance qu'on apporte aux postes dans le LAN, il peut être nécessaire de filtrer le traffic sortant vers internet. Mettre en place un filtrage autorisant le LAN à n'accéder qu'aux ports HTTP et HTTPS en TCP, ainsi que l'ICMP. Ne pas oublier d'ouvrir aussi le port DNS.. Tester chacune des regles indépendamment en ouvrant une page web, avec un ping, et la commande `nslookup` pour tester la résolution des noms.

Filtrage simple des connexions bruteforce

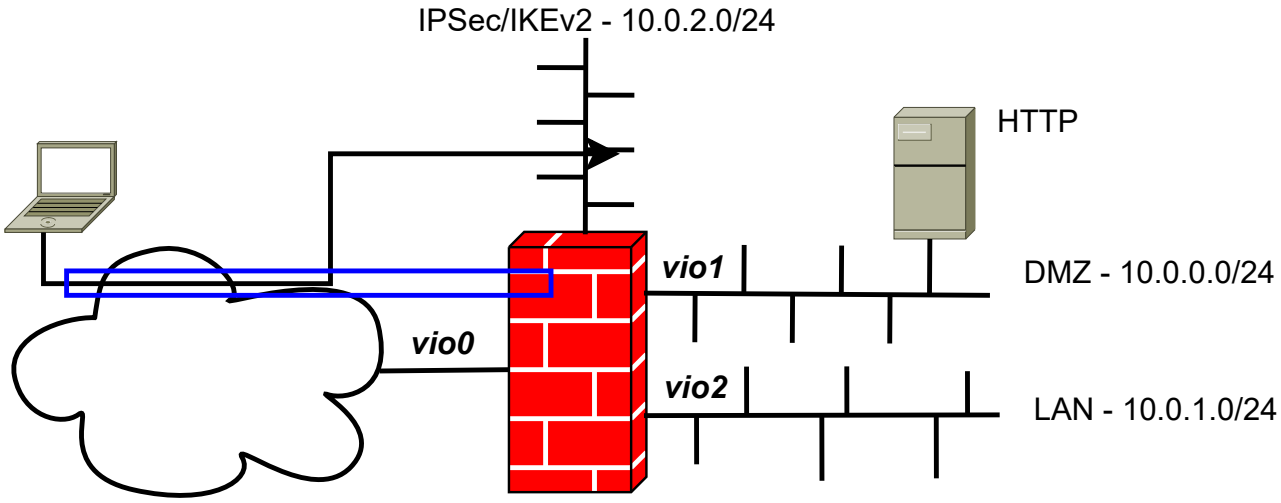
En utilisant la directive `overload` , modifier la régle autorisant les connexions SSH pour autoriser les connexions depuis n'importe quelle provenance, mais en blacklistant l'IP source après plus de 3 tentatives de connexion en 60 secondes. Faire blacklister volontairement une IP de la salle, observer la table PF correspondante, et enlever le blacklistage de cette IP.

> Comment faire pour s'assurer que certaines IP ne seront jamais blacklistées ?

Partie II : Le VPN

Preambule

Le VPN est un élément clé de tout réseau d'entreprise, il permet de relier divers sites distants, de donner accès à des ressources privilégiées à des utilisateurs "sur la route", et de manière générale il se doit d'être chiffré pour d'évidentes raisons de confidentialité, et authentifié pour des raisons de sécurité.



Il existe plusieurs types de VPN, les plus connus sont:

- IPSec, permet de faire du VPN site-to-site, et de connecter des clients distants. Est en général implémenté directement dans la pile IP du noyau.
- OpenVPN se base sur SSL/TLS pour le chiffrement, l'échange de clefs/authentification. Utilise un protocole maison non standardisé, tourne en espace utilisateur, et nécessite un client spécifique. Il est aussi largement déployé.

Pour l'exemple que nous allons mettre en place avec iked(8), il est conseillé de:

- noter les configurations réseau/tables de routage
- tester la connectivité via ping, la résolution DNS via `nslookup`
- faire une connexion avec nom d'utilisateur/mot de passe en clair (telnet/FTP) et écouter le traffic via `wireshark` sur la machine hôte, simulant ainsi un attaquant potentiel
- refaire la même chose via le VPN pour s'assurer que l'on ne voit plus passer le traffic en clair

Dans cette partie, uniquement le firewall et une VM windows sont utilisés.

IKEv2

OpenBSD fournit un démon IKEv2 dans le système de base nommé iked(8), qui se configure via le fichier iked.conf(5).

La commande ipsecctl(8) permet de voir les flux et associations configurés dans le noyau.

Une page de la FAQ est dédiée à son utilisation, et peut servir de base à notre configuration. Il est **fortement conseillé** de lire cette page.

Utilisation de ikectl

iked(8) nécessite une architecture à clef publique/clef privée, vous pouvez utiliser la sous-commande `ca` de ikectl(8) pour créer rapidement une CA (Authorité de Certification), puis un couple clef/certificat pour le serveur et le client.

Assurez vous tout d'abord d'avoir une heure correcte (potentiellement avec rdate(8), en utilisant le serveur `pool.ntp.org`) pour ne pas génerer de certificat/clef ayant une date de validité dans le futur..

> Noter les fichiers générés par chacune des étapes, ainsi que leurs droits d'accès.

Configuration serveur OpenBSD

Configurer un repondeur passif permettant à un client VPN d'accéder au sous-réseau de la DMZ, en utilisant l'authentification par login/mot de passe CHAPv2, et distribuant des IPs dans le sous-réseau 10.0.2.0/24.

Démarrer le VPN, et s'assurer qu'il fonctionne correctement en observant les fichiers journaux du système, la liste des interfaces, ainsi que les serveurs en écoute.

Activer iked(8) au démarrage de la VM.

> Quelles sont les régles de firewalling à ajouter pour qu'un client puisse se connecter au VPN ? Quels sont les fichiers de clefs de la PKI qui vont être utilisés par le serveur ? Quelles sont les IP utilisées par le VPN ?

Configuration client Windows

Dans Virtualbox, reconfigurer l'interface de la machine virtuelle Windows pour qu'elle soit en mode *pont* sur `eth0` . Vérifier qu'elle récupère bien une adresse via DHCP depuis le serveur de la salle. A ce stade, elle est vue comme étant "sur internet" donc dans un environnement potentiellement hostile. Elle n'a plus tout accès aux ressources fournies par le LAN ou la DMZ.

A ce stade la, il faut faire parvenir au client les certificats nécessaires pour l'établissement de la connexion VPN. Cette transmission doit idéalement se faire via un canal sécurisé (SFTP via WinSCP, mail crypté via GPG...). Pour le TP, on fera avec les moyens du bord (serveur HTTP, clef USB...).

> Quels sont les fichiers/certificats transmis au client ? Quels sont les paramêtres à utiliser pour la configuration du client ?

Démarrer le VPN coté client - si tout s'est bien passé, la connexion est établie. Faire divers tests, et observer ce qu'il se passse des deux côtés.

Route par défaut ?

On peut utiliser un VPN selon 2 modes:

- uniquement pour le traffic à destination des réseaux privés du coté du serveur. Tout le reste du traffic passe par la connexion internet existante, et n'est pas crypté.
- tout le traffic passe par le VPN. C'est la configuration par défaut du client IKEv2 de windows. Comment modifier ce comportement ?

> Comment voir/debugguer le traffic passant à travers le VPN ? Quels sont les avantages/inconvénients des deux méthodes ? Comparer l'utilisation du réseau en utilisant les deux méthodes. Quelles sont les différentes méthodes possibles pour que les réseaux 10.0.0.0/24 et 10.0.2.0/24 se "voient" ? Que faut il configurer sur le firewall pour qu'un client VPN puisse accéder à internet ? Tenter une connexion SSH depuis le client VPN vers le serveur dans l'autre sous-réseau.

Jonction de deux sites

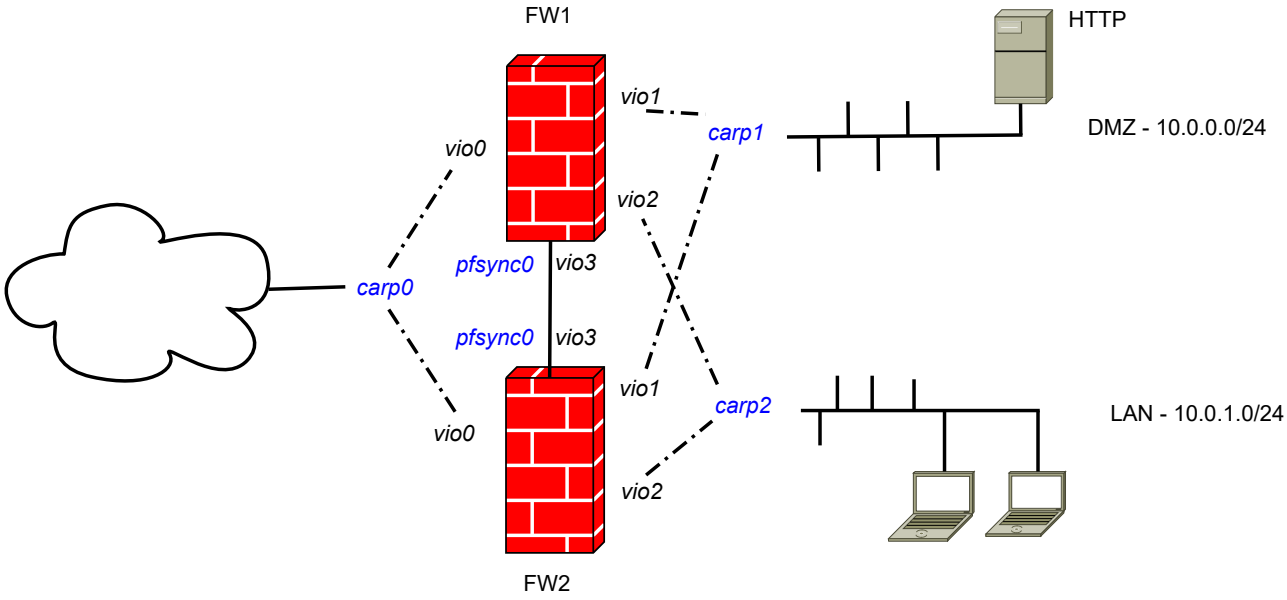
Reprendre la configuration existante, et configurer un VPN entre deux sites distants (avec celui du binôme voisin par exemple). Evidemment, il faut renuméroter les réseaux pour éviter les conflits...

Partie III : CARP/Pfsync

Généralités

C'est bien d'avoir un firewall pour protéger son réseau, mais comme tout élément d'infrastructure critique, c'est encore mieux de le doubler pour éviter qu'il représente un SPOF. Le fait d'avoir une infrastructure redondante en tout point présente plusieurs avantages :

- si un des firewall tombe pour une raison X ou Y, son jumeau prend automatiquement le relais. C'est une configuration nommée **failover** ou tolérante aux pannes
- lorsque les deux sont en fonctionnement, ils peuvent aussi être configurés pour se répartir automatiquement la charge, dans une configuration en **load-balancing**
- on peut éteindre un des deux firewall pour faire de la maintenance ou une mise à jour sans occasionner de coupure de service, la bascule se faisant de manière totalement transparente pour les utilisateurs



Cette configuration se fait sous OpenBSD avec une interface virtuelle particulière nommée carp(4) supportant une IP virtuelle partagée par un groupe de machines, et une deuxième interface de type pfsync(4) va être utilisée pour synchroniser les états des connections actives entre les deux firewall, afin que la bascule se fasse sans coupure de connection. Pour les besoins de notre TP, nous allons cloner la machine virtuelle OpenBSD (on aura donc fw1 et fw2, ne pas oublier de réinitialiser les adresses MAC, changer son hostname, et de supprimer les clefs ssh dans /etc/ssh/ pour éviter des alertes de ssh - elles seront regénérées au démarrage de la VM), et modifier chacune des configurations :

- dans VirtualBox, activer une 4e interface physique (de type **virtio-net**) dans la configuration réseau: elle servira de support a pfsync(4). Configurer les 2 interfaces pour qu'elles soient dans le même réseau interne distinct de `lan` et `dmz` pour simuler une connection directe entre les 2 machines.
- dans VirtuelBox, choisir 'tout autoriser' pour le mode promiscuité des 3 premières interfaces. C'est nécessaire pour le bon fonctionnement de carp dans virtualbox

- dans OpenBSD, modifier les configurations des deux firewall pour que chacune des interfaces 'clones' se retrouvent dans le même sous-réseau.

    - `vio0` reste en DHCP sur fw1 et fw2, qui doivent donc chacun récupérer une IP publique distincte
    - `vio1` sur fw1 aura l'addresse `10.0.0.100` et `10.0.0.200` sur fw2
    - `vio2` sur fw1 aura l'addresse `10.0.1.100` et `10.0.1.200` sur fw2
    - `vio3` sur fw1 aura l'addresse `10.0.10.1` et `10.0.10.2` sur fw2, sur un sous-reseau en /30 (c'est l'interface pour pfsync(4))

Failover

Il faut tout d'abord activer CARP via sysctl(8), et s'assurer que le changement est permanent via sysctl.conf(5). Ne pas oublier d'activer la préemption pour que la bascule se fasse automatiquement.

Avec ifconfig(8) nous allons créer 3 interfaces virtuelles carp(4) pour chacun des sous-réseaux utilisés par nos firewall, en utilisant une passphrase distincte pour s'assurer qu'un intrus ne puisse pas forcer la bascule et perturber le fonctionnement du réseau. Ces interfaces vont supporter l'ip qui était précedemment assignée aux interfaces physiques de fw1 (qui restera pour l'instant le maitre, fw2 servant uniquement de backup). Attention a ne pas oublier le paramêtre `advskew` sur fw2 pour bien lui spécifier qu'il est en mode BACKUP.

> Comment est utilisée la valeur fournie via `advskew` ?

Il faut (sur fw1 et fw2) :

- pour `carp0` , lui assigner une IP libre sur le sous-réseau en DHCP de la salle, en utilisant un `vhid` **unique** pour éviter les conflits dans la salle - c'est l'identifiant de réseau virtuel. On peut utiliser le dernier chiffre de l'IP publique choisie.
- pour `carp1` , lui assigner l'ip qui était auparavant sur `vio1` , donc 10.0.0.1/24 des deux cotés avec le `vhid 1` . Ici, pas de conflit possible dans la salle, chacune des interfaces est dans son propre sous-réseau
- pour `carp2` , lui assigner l'ip qui était auparavant sur `vio2` , donc 10.0.1.1/24 des deux cotés avec le `vhid 2` .
- configurer une interface `pfsync0` utilisant `vio3` comme support.

Observer la configuration de chacune des interfaces, et la rendre permanente via un ensemble de fichiers hostname.if(5) A ce stade, la synchronisation ne se fait pas, car pf bloque le protocole CARP.

Il reste a configurer PF pour autoriser le traffic du protocole carp sur les interfaces `vioX` (utiliser un groupe d'interface) et d'ignorer l'interface `pfsync0` dans les regles de filtrage (utiliser `set skip`)

Du point de vue de PF, le traffic arrive toujours sur les vraies interfaces, donc on n'a pas besoin de beaucoup modifier la config. Ne pas oublier de changer l'ip d'écoute du VPN dans iked.conf(5), sasyncd(8) si on veut du failover entre les deux instances de iked.

> Que faut il faire pour que le VPN marche sur l'ip flottante ? Est-ce que le VPN peut gérer le failover ?

A ce stade, les clients connectés sur les réseaux `dmz` et `lan` doivent toujours avoir acces à internet - adapter les rêgles PF si besoin. Vérifier que les états des connections sont bien synchronisés en observant `systat states` sur chacun des firewall, en simulant des connections depuis les deux réseaux.

> Certains états ne devraient pas être synchronisés, lesquels, pourquoi, et comment s'assurer qu'ils restent locaux a chacun des firewall ?

éteindre fw1, et s'assurer que la bascule se fait bien sur fw2, les connections actives devant rester utilisables.

> Faut t'il modifier la rêgle de NAT pour les différents réseaux ? Quels sont les conséquences sur les paquets en provenance des sous-réseaux ?

Load Balancing

CARP peut aussi être utilisé pour faire de l'équilibrage de charge entre les deux firewall, les connections passeront de manière égale via les deux firewall. La page de manuel de carp(4) explique comment configurer les interfaces pour configurer ce mode. Faire des tests avec plusieurs clients, et sur les différents modes IP et ARP balancing. Observer la distribution des connections avec *systat(1) rules*.

Partie IV : Reverse Proxy

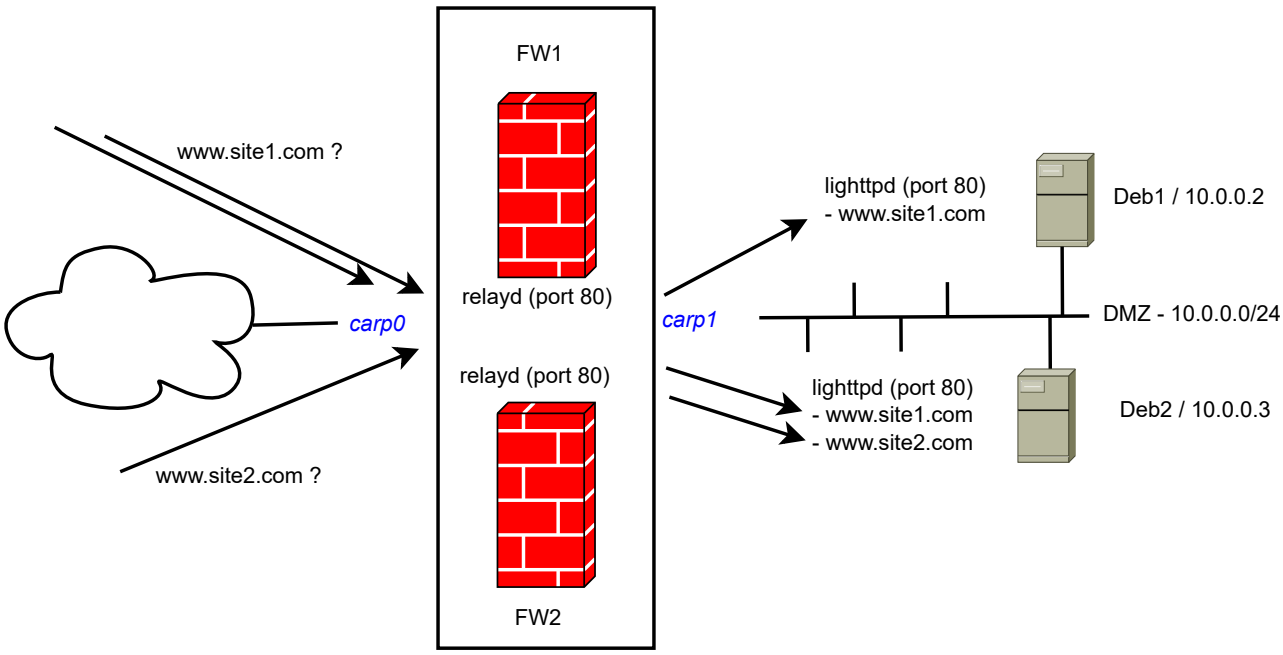
Généralités

On parle de reverse-proxy quand on reçoit une connection d'un client a destination d'un serveur que l'on contrôle sur une des nos IPs publiques, connection qui va être filtrée/redirigée/modifiée en fonction de l'état du/des serveurs visés. Pourquoi reverse-proxyfier ? Les raisons peuvent être multiples:

- avoir plusieurs sites sur des serveurs distincts mais tous accessibles sur la même addresse IP
- faire du load balancing entre plusieurs serveurs backends hébergeant le même site
- faire du failover applicatif si un serveur ne répond plus (ie renvoyer une page d'erreur)
- rediriger le trafic vers un serveur fonctionnel pendant une maintenance planifiée
- faire de la désencapsulation/accéleration TLS: La connection TLS (coûteuse en temps de traitement) est terminée au niveau du reverse-proxy, et la communication avec les backends se fait en clair
- faire du filtrage basique en fonction de l'adresse demandée par le client ou de certains entêtes
- peut aussi faire du filtrage basique sur header/URL
- faire de l'interception TLS pour inspecter le traffic sortant sur HTTPS. C'est un cas très particulier..

Certaines de ces choses sont possibles nativement avec pf(4), mais ce dernier ne permet pas de vérifier la disponibilité du/des serveurs cibles, il s'occupe uniquement de transmettre la connection.

Un serveur web tel que `nginx` ou `apache` peut aussi faire reverse-proxy, mais dans notre cas OpenBSD fournit un démon spécifique plus flexible nommé relayd(8), qui se configure via relayd.conf(5) et peut se contrôler a la volée via relayctl(8). Il s'integre étroitement avec pf(4) et va lui rajouter des régles via un système d'ancres.



Proxy niveau 3 / redirection web avec loadbalancing

Dans cette première partie, nous allons mettre en place un reverse-proxy dit 'redirecteur' faisant du load balancing pour distributer les requêtes des client vers un ensemble de serveurs, en s'assurant que les dits serveurs sont en état de fonctionnement correct et que les connections sont bien réparties.

Dans ce cas, le reverse proxy va périodiquement interroger les serveurs pour vérifier qu'ils répondent bien, et rediriger via PF les requètes des clients directement vers les serveurs.

Pour avoir deux serveurs web, tout d'abord il vous faut cloner la machine virtuelle debian.

Ne pas oublier de les différencier en reconfigurant leurs hostnames et adresse IP (mettre le clone en `10.0.0.3` , supprimer les clefs ssh identifiant la machine et les recréer (cf ssh-keygen(1)) et le rendre accessible depuis l'extérieur via ssh sur le port 2223 des firewall).

> Quels sont les fichiers a remodifier sur le clone ? Vérifier que les deux firewall peuvent y accéder.

Modifier le contenu de la page HTML servie par défaut pour savoir quel est le serveur répondant à une requête.

Ne pas oublier d'activer le logging des requêtes dans les configurations de lighttpd (`lighttpd-enable-mod accesslog`). Suivre le log via `tail -f /var/log/lighttpd/access.log` .

La suite des étapes peut se faire au choix sur les deux firewall ou uniquement sur un des deux, il faut juste s'assurer qu'on interroge bien le bon firewall si uniquement un des deux est utilisé (dans le cas ou on a configuré CARP en mode load-balancing) et ne pas oublier de synchroniser les configurations des instances de `relayd(8)` .

Configurer une redirection dans `relayd(8)` pour qu'il distribue les requêtes sur le port 80 sur les deux serveurs web, en mode round-robin (les requêtes vont arriver alternativement sur les deux serveurs. Quels sont les autres modes ?).

> Que faut t'il configurer dans `pf(4)` pour que relayd puisse fonctionner correctement ? Observer les règles PF et les tables.

Observer les logs de `relayd(8)` dans `/var/log/daemon` , et utiliser `relayctl show summary` pour avoir une vue d'ensemble de l'état des relais/redirections/backends.

Vérifier que l'équilibrage se fait bien en rechargeant plusieurs fois la page web depuis le poste hôte, et en observant les logs des deux serveurs web.

Failover

Eteindre l'un des serveurs, observer ce qu'il se passe dans les tables PF, et continuer à envoyer des requêtes.

Configurer relayd pour qu'il interroge une page donnée sur le serveur web et vérifie la somme de contrôle de cette page avant de forwarder la requête.

> Modifier le contenu d'une page pour tester ce contrôle. A quoi sert ce mode ?

Relayd peut en cas de panne des deux serveurs renvoyer vers un serveur 'alternatif' présentant une page d'erreur. Configurer le serveur httpd(8) (fourni dans l'OS de base) directement sur le firewall avec une page web par défaut, et configurer relayd(8) pour l'utiliser en cas de panne. Eteindre le 2e serveur web, et continuer d'envoyer des requêtes. Observer les tables et régles PF, et rallumer progressivement les deux serveurs web tout en observant le comportement de PF.

Reverse proxy simple relai niveau 7

Dans cette partie, nous allons mettre en place un reverse-proxy dit *applicatif* inspectant la requête du client, et l'aiguillant en fonction vers le bon serveur.

Dans ce cas, c'est le reverse proxy lui-même qui ouvre la connection vers le serveur final, va faire la requête, et transmettre le résultat au client qui attend sur la connection que ce dernier a ouvert avec le reverse-proxy - mais c'est transparent pour le client.

Rajouter un virtual host uniquement sur un des deux serveurs web, et configurer relayd(8) pour renvoyer les requêtes à destination de ce virtualhost vers le bon serveur.

> Attention, pour `relayd` ce n'est plus une redirection niveau 3, mais un relai niveau 7. Pourquoi ?

> Dans chacun des cas, quelle est l'ip cliente enregistrée dans le log de serveurs web ? Que faut t'il faire dans la configuration de relayd pour que le serveur web ait une information correcte ?

Partie V : Proxy sortant

Généralités

On parle de proxy sortant quand on reçoit la requête d'un client interne à notre réseau à destination d'un serveur web à l'extérieur, et on ne la transmet pas directement à la cible - on va d'abord l'analyser pour savoir si cette requête est légitime, ou cacher éventuellement la réponse pour la resservir à un autre futur client.

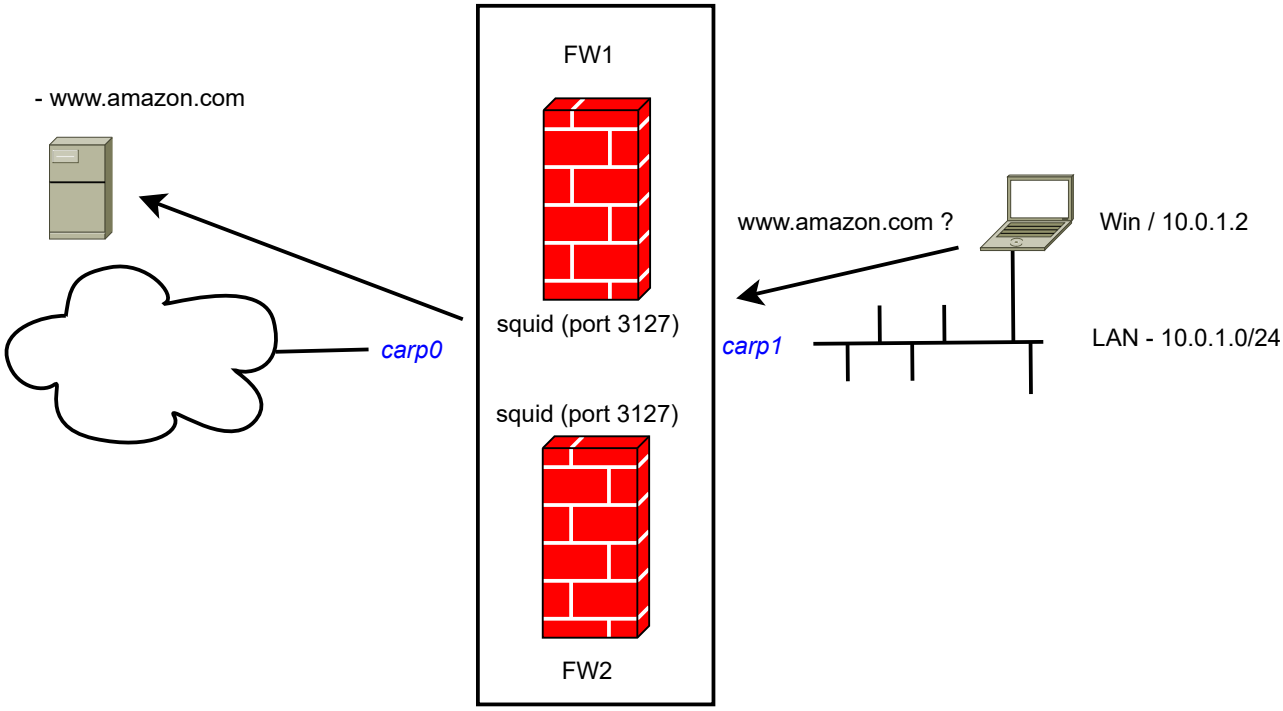
Pourquoi proxyfier ? Pareil, ici les raisons peuvent être multiples:

- filtrer sur les contenus (illégaux, contre-productifs..) en les analysant en profondeur
- interdire l'accès à des adresses/domaines complets
- mettre en cache le contenu fréquemment demandé par les clients pour diminuer le traffic
- faire des statistiques sur l'usage du web de nos utilisateurs
- réécrire/filtrer des entêtes pour anonymisation des utilisateurs
- faire de l'inspection de trafic TLS (toujours un cas particulier!)

Un proxy peut-être configuré de deux manières principales :

- en mode transparent/interception: l'utilisateur ne sait pas qu'il passe par un proxy, c'est le firewall qui détourne les requêtes du client sur le port 80 de manière transparente vers le proxy
- en mode classique: le navigateur/poste client doit être configuré pour utiliser le proxy sur un couple adresse:port connu. Dans ce cas, le proxy peut demander une authentification au client pour faire du suivi. Généralement dans ce cas le firewall bloque complètement l'accès au port 80 en sortant pour obliger tout les clients à passer par le proxy.

Dans cette partie nous allons successivement utiliser deux proxies: tout d'abord tinyproxy pour faire un proxy filtrant simple, puis squid en proxy cache transparent.



Proxy filtrant

Installer `tinyproxy` (toujours via pkg_add(8)), et le configurer pour bloquer l'accès à facebook/gmail/youtube/isima.fr. Activer le logging, et anonymiser les entêtes envoyés par les navigateurs.

Reconfigurer le navigateur dans les clients du LAN pour utiliser l'ip et le port du proxy (pour tout les protocoles y compris ssl).

> Quelle IP:port utiliser ?

Ne plus NATer les port http et https pour le LAN.

> Y'a t'il quelque chose d'autre à faire sur les firewalls ?

A partir de ce stade, toutes les connections depuis le navigateur devraient passer par le proxy.

Suivre `tinyproxy.log` en utilisant `tail -f` , et juste démarrer firefox sur un client. Que se passe t'il ? Naviguer sur quelques pages, essayer d'aller sur des pages interdites, et aller consulter les statistiques du proxy sur son interface web.

Jouer sur les réglages d'anonymisation des entêtes, et observer le traffic avec `wireshark` ou `tcpdump -A` . Visiter amiunique.org pour se faire une idée de la différence..

> Quel sont les problèmes posés une blacklist basée sur les domaines ? Avec un grand nombre de clients et de connections, quel est l'autre problème qui peut subvenir sur les firewall ?

Proxy cache

Sur OpenBSD, une fois installé squid vient avec une documentation aidant à le configurer rapidement - consulter `/usr/local/share/doc/pkg-readmes/squid` et `/usr/local/share/examples/squid/squid.conf.documented` .

Pour configurer squid en mode transparent, il faut rediriger toutes les connections provenant des sous-réseaux sur le port 80 vers le port ou squid écoute.

> Quelle est la directive de `pf.conf` pour ceci ? Quelle est la directive correspondant dans `squid.conf` ?

Squid permet de cacher le contenu, ce qui permet de diminuer la bande passante utilisée globalement par tout les clients en resservant le contenu déja consulté et non expiré.

Configure un cache local sur disque, l'initialiser et démarrer squid. Observer les logs dans `/var/squid/logs/` .

Sur un pc client (sans avoir de proxy configuré dans le navigateur), essayer d'accéder à une page web - observer les logs de squid, et le contenu de /var/squid/cache - recharger la page plusieurs fois, que se passe t'il ?

Configurer squid pour pouvoir accéder à l'interface web *manager* depuis un poste choisi. Observer les valeurs au fur et à mesure de l'utilisation du proxy par les clients.

Configurer `squid` pour pouvoir purger des objets du cache à la main en utilisant `squidclient` . Attention les objets doivent être purgés un par un, on ne peut pas directement supprimer un site/page complet.

> Quelle configuration spécifique faut t'il faire pour pouvoir aussi proxifier les requètes en https ? Est-ce compatible avec le mode de fonctionnement transparent ? Quels sont les nouveaux problèmes posés ?

> Que pourrait on faire de plus en complétant squid avec squidguard ?

Partie VI (Bonus) : SSH

SSH est un outil réseau extrêmement puissant. Nous allons rapidement voir quelques fonctionnalités cruciales dans la gestion sécurisée d'accés distants sur des réseaux. Tous ces tests peuvent se faire depuis la machine hôte vers le firewall, ou depuis le firewall vers le serveur SMTP, ou même depuis la machine hôte vers le serveur SMTP en traversant le firewall.

Ce wikibook est une très bonne ressource sur les fonctionalités de SSH.

Clefs SSH

Se créer une clef SSH avec une passphrase, interdire l'accès par mot de passe sur le serveur, et se connecter en utilisant la clef que l'on vient de créer.

> Quelles sont les commandes correspondantes ?

En utilisant ssh-agent(1), enregistrer sa clef pour ne plus avoir à retaper sa passphrase.

> Comment lister les clefs connues de l'agent, et supprimer une clef ?
>
> Comment configurer le serveur pour n'autoriser que certaines clefs à se connecter à un compte donné ?
>
> Comment configurer le serveur pour n'autoriser qu'un certain groupe UNIX à se connecter via une clef ?
>
> Comment utiliser la même fonctionnalité pour n'autoriser que certaines commandes à être lancées ?

Mettre en place une tâche de synchronisation `rsync(1)` avec une clef sans passphrase pour illustrer ce fonctionnement.

> Quels sont les atouts et inconvénients ? Quels peuvent être les problèmes posés par l'utilisation de telles clefs ?

Réutilisation de connexion

En utilisant une fonctionnalité de SSH, comment faire pour se connecter de multiples fois au même serveur en ayant à taper son mot de passe qu'à la première connexion ?

Traversée de firewall

Il existe 2 techniques utilisant des fonctionnalités de SSH pour traverser un firewall (ie se connecter à une machine étant derrière le firewall) en ayant un compte sur ce firewall. Les illustrer en se connectant en ssh de la machine hôte vers le serveur SMTP, et en faisant un renvoi de port sur le firewall se connecter directement au serveur web sur le port 80 depuis la machine hôte.

SFTP

Configurer le serveur SSH pour autoriser un groupe d'utilisateur UNIX à se connecter uniquement via SFTP, en s'assurant que les utilisateurs ne peuvent pas sortir de leur répertoire personnel. Comment tester/afficher la configuration du serveur pour un utilisateur donné sans faire de connexion ? Tester ensuite directement via un client tel que Filezilla ou

lftp. Autoriser un second groupe d'utilisateur à se connecter uniquement via SFTP, et uniquement avec une clef.

> Quels sont les avantages de SFTP par rapport à un serveur FTP classique ? Et les inconvénients ?

VPN

En utilisant uniquement SSH, créer un VPN entre deux machines.

> Quels sont les avantages et inconvénients de cette technique ?