

Licence Professionnel ASRALL

Projet tuteuré.



Ufwi

Groupe :
– Cyril PIERRÉ
– Maxime ROBIN
– Valentin FROLICH
– Simon BARROT

Sommaire

1	Introduction	3
1.1	Historique	3
1.2	Présentation	3
2	Liste des solutions de pare-feu par identification	5
3	Externalisation des logs dans une BD MySQL	7
4	Fonctionnement d’ufwi	9
4.1	Algorithme	9
5	Daemon ufwi-authd	11
5.1	Introduction	11
5.2	Intallation	12
5.3	Quelques commandes liées au daemon	12
5.4	Fichier de configuration	13
5.5	Installation des certificats	13
5.6	Configuration basique du daemon	14
6	Daemon ufwi-filterd	15
6.1	Introduction	15
6.2	Intallation	15
6.3	Compilation	16
6.4	Commandes	16
6.5	Fichier de configuration	16
7	Daemon ufwi-rpcd	17
7.1	Introduction	17
7.2	Installation	17
7.2.1	Pré-requis	17
7.3	Utilisation	18
7.3.1	Démarrage	18
7.3.2	ufwi_rpcd_client (nucentral_client)	18
8	Avancement du projet à mis parcours	20
9	Difficulté rencontré et avis personnel	21

1

Introduction

1.1 Historique

Il faut savoir que le projet Ufwi descend du projet NuFW. La première version publique de NuFW est sortie le 01 septembre 2003. L'idée avait germé en 2001 mais ne s'était concrétisée que le 01 septembre de l'année 2003. La société croît rapidement, créant une vingtaine d'emplois en région parisienne, et acquérant une réputation dans le milieu de la sécurité informatique. En 2009, l'entreprise décide d'abandonner le pôle service pour se concentrer sur l'édition d'un pare-feu clef en main, EdenWall, couplant NuFW avec une interface d'administration performante, cette action entraîna des difficultés financières lourdes à la société. Le 18 août 2011, soit sept ans après sa création, le tribunal du commerce de Paris a prononcé la liquidation de la société, de ce fait la fin du projet NuFW.

Cependant le projet fut repris sous le nom de Ufwi, offrant une reprise optimale du projet en opérant un rassemblement du code, notamment éparpillé chez les clients de la société, qui évite les mois de travail nécessaires à la récupération des modifications ayant eu lieu depuis cette date.

1.2 Présentation

Ufwi est une solution de pare-feu par authentification, cela signifie qu'il effectue une authentification de chaque connexion qui le traverse. L'authentification se fait de façon transparente, en requérant les informations d'identification via l'annuaire des utilisateurs d'un réseau (LDAP ou autre). Donc avec Ufwi chaque utilisateur doit décliner son identité à chaque initialisation de connexion. Une fois que l'utilisateur est identifié, le paquet se voit filtré, selon les droits associés à l'utilisateur. Une fois que le premier paquet est accepté, les paquets suivants appartenant à la même connexion sont gérés par le système de suivi d'état (seul le paquet d'initialisation d'une connexion est authentifié). Ce principe de fonctionnement requiert la présence d'un client sur le poste utilisateur, il existe des clients compatibles pour Windows et GNU/Linux. Ufwi dispose aussi d'un mode sans client, pour se faire Ufwi utilise un module "ufwi-authd" (utilisé dans le cadre de notre projet).

Comme vu précédemment Ufwi, chaque connexion est associée à un utilisateur. Ce qui veut dire que plusieurs utilisateurs peuvent travailler sur un même poste génèrent simultanément des flux à partir d'une même adresse IP source.

Donc Ufw permet le filtrage par utilisateur mais peut aussi apporter d'autres fonctionnalités intéressantes.

- Ufw permet de contribuer de manière très pointue à la surveillance de l'activité réseau des serveurs. En règle générale dans les installations modernes on attribue des utilisateurs distincts à chaque service que fait tourner le serveur, de ce fait avec Ufw on peut attribuer une politique réseau différente pour chaque utilisateur système.
- Ufw peut marquer chaque paquet d'une connexion avec les informations de son utilisateur (identifiant...) ce qui permet d'appliquer une politique de qualité correspondant à chaque utilisateur. Cela permet donc de distribuer la bande passante entre chaque utilisateur, ainsi attribuer une bande passante plus élevée au utilisateur qui utilise des applications plus gourmandes en termes de connexion.
- Ufw comprend un module de surveillance qui journalise les événements principaux se produisant sur le réseau en indiquant les utilisateurs à l'origine de ses flux (ouverture, fermeture de connexion, paquet bloqué, etc...). Ces logs peuvent être gérés au choix par Syslog, PostgreSQL ou MySQL. Ufw conserve chaque connexion ouverte (ou tentée) même si l'utilisateur en question a changé d'adresse IP ou de machine.
- Ufw permet d'identifier les utilisateurs des applications réseau grâce à une authentification unique (Single sign on). Grâce au système de journalisation SQL une table de connexion est maintenue en temps réel.

2

Liste des solutions de pare-feu par identification

Les pare-feu par identification les plus connus :

- AuthPF : Fonctionne sous OpenBSD et qui se repose sur SSH pour l'identification des utilisateurs : <http://www.openbsd.org/faq/pf/authpf.html>
- NuFW : projet ayant donné naissance à UFWI suite à la liquidation de l'éditeur "Eden-Wall Technologies"
- Cyberoam : pare-feu entièrement basé sur l'identification, en utilisant une corrélation entre adresse MAC et utilisateur : <http://www.cyberoam.com/fr/firewall.html>
- CheckPoint (NAC Blade) : utilisation des règles de filtrage en fonction d'une authentification basée sur Kerberos, l'identité de son poste et du niveau de sécurité du poste (mise à jour de sécurité / antivirus) : <http://www.cyberoam.com/fr/firewall.html>

Ci-dessous un tableau comparatif des différent avantage et inconvénient des différentes solution de pare-feu par authentification :

Solution parefeu	Avantage	Inconvénient
Ufwi	<ul style="list-style-type: none"> – Projet devenu libre. – Appliquer des règles horaires strictes. – composé de deux démons qui peuvent être mis en place sur des systèmes différents et le démon principal est massivement multithreadé. – Contrôle d'accès sont réalisées grâce à des greffons (des modules system, ldap, dbm, plaintext). – Journalisation de l'activité des utilisateurs peut être faite par syslog, mysql, postgresql. – Multi plate-forme. – Génération des acl grace a un module (nuaclgen) 	<ul style="list-style-type: none"> – Tres peu de documenation voir aucune (obligé de cf. à nufw). – Nécessite une partie cliente spécifique sur les postes clients. – Payante sous windows (NuWinC). – Projet peu soutenu (temps de reponse 20-30 jours).
AuthPF	<ul style="list-style-type: none"> – règles par système et/ou par utilisateur. – Connection via SSH. – chaque utilisateur peut disposer de ses propres règles, présentes dans HOME/.authpf/authpf.rules. – Simple a metre en place. – C'est un Firewall de type filtre de paquet (Couches réseau et transport). – Documentation et suport en français et assez simple. 	<ul style="list-style-type: none"> – l'utilisateur se déconnecte, le pare-feu lui retire les règles ajoutées. – quelqu'un d'autre peut toujours spoofer cette IP et passer. – Necessite la creation d'un home pour chaque utilisateur. – Necessite Openbsd et des conaissance PF.
Cyberoam	<ul style="list-style-type: none"> – N – A 	<ul style="list-style-type: none"> – N – A
CheckPoint	<ul style="list-style-type: none"> – N – A 	<ul style="list-style-type: none"> – N – A

3

Externalisation des logs dans une BD MySQL

Configuration du serveur BD

Installation des paquets :

```
apt-get install apache2 php5 mysql-server nolog
```

Configuration de la passerelle :

Configuration IP :

```
ifconfig eth0 192.168.1.137/24 ifconfig eth1 172.20.8.1/24
```

Installation des paquets :

```
apt-get install ulogd ulogd-mysql
```

Correction d'un bug : ajout d'une ligne dans le script de démarrage qui va charger un module

```
nano /etc/init.d/ulogd export LD_PRELOAD = /usr/lib/libmysqlclient.so.16
```

Configuration de ulogd : modification de son fichier de configuration

```
nano /etc/ulogd.conf
```

Décommenter la ligne 46 (pour charger un module supplémentaire)

Renseigner les informations de connexion à la base de données :

```
paragraphe «[MYSQL]» ligne 59 : table="ulog" pass="passulog" user="ulog" db="ulog" host="172.20.8.2"
```

Configuration du serveur de BD :

Configuration IP :

```
ifconfig eth0 172.20.8.2/24
```

Lister tous les fichiers installés à l'installation de nolog :

```
dpkg -L nolog | more
```

Ouvrir le fichier suivant (démarche à suivre pour créer les tables de la base de données)

```
nano /usr/share/doc/nolog/README.Debian
```

Connexion à la base de données et création de l'utilisateur (les deux programmes vont se connecter avec ce compte) :

```
mysql -u root -p create database ulog; create user 'ulog'@'%' identified by 'passulog'; grant all privileges on ulog.* to ulog; exit
```

Commandes de création de la base :

```
cd /usr/share/doc/nolog/scripts gunzip ipv4.sql.gz cat ipv4.sql | mysql -u ulog -p ulog
```

Modification du fichier de configuration de mysql

```
nano /etc/mysql/my.cnf ligne 47
```

Il faut qu'il écoute sur l'interface 172.20.8.2

```
bind address= '172.20.8.2'
```

Renommer les fichiers de configuration :

```
cd /etc/nulog cp default.core.conf core.conf cp default.nulog.conf nulog.conf cp default.wrapper.conf wrapper.conf
```

Renseigner les informations de connexion à la base de données :

```
nano core.conf host=localhost db=ulog user=ulog password=passulog table=ulog
```

Prise en compte des changements : redémarrage de services Sur la passerelle :

```
/etc/init.d/ulogd restart
```

Sur le serveur :

```
/etc/init.d/ulogd restart
```

On choisit ce que l'on veut loguer avec iptables

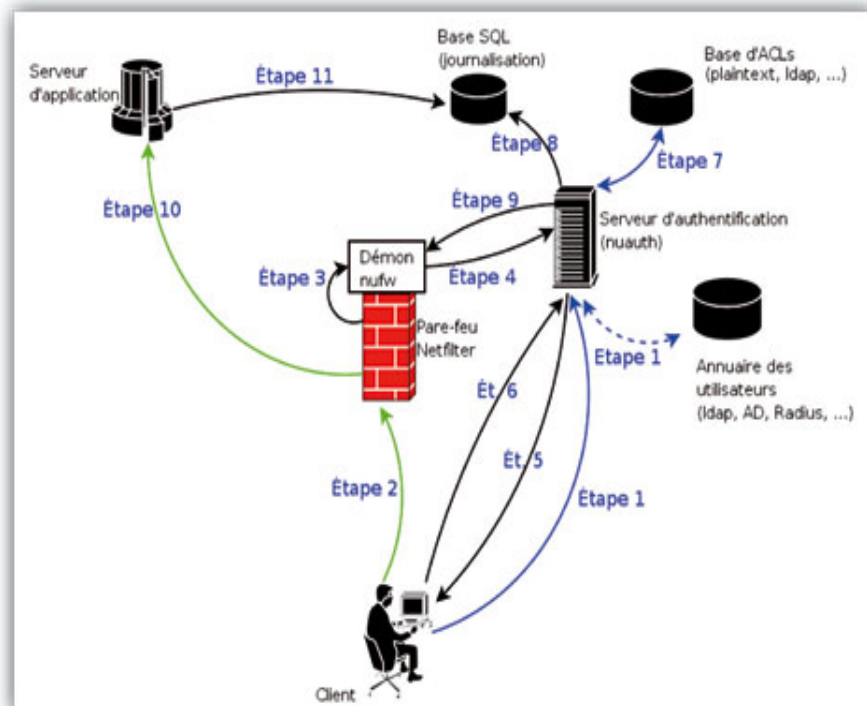
4

Fonctionnement d'ufwi

4.1 Algorithme

1. Au démarrage de sa session de travail, l'utilisateur lance un client sur son poste de travail. Le client ouvre un tunnel crypté par TLS vers le serveur nuauth qui réalise l'authentification de l'utilisateur pour ce tunnel au regard d'un annuaire référent. Ce tunnel est ensuite conservé pendant toute la durée des opérations et l'ensemble des échanges entre nuauth et le client se fait ensuite par son intermédiaire.
2. Supposons maintenant que l'utilisateur ouvre une connexion vers un serveur, par exemple une connexion web à destination du serveur d'application. Il utilise alors son navigateur favori qui envoie un paquet à destination du serveur. Le paquet envoyé est un paquet TCP standard avec notamment comme caractéristiques port destination 80 (web) et bit SYN positionné (ouverture de connexion).
3. Le pare-feu (Netfilter) envoie ce paquet au démon nufw (au moyen de la décision QUEUE ou NFQUEUE) : contrairement à un pare-feu " classique ", Netfilter ne prend pas de décision à propos de cette connexion.
4. Le démon nufw se contente de relayer le paquet reçu vers le serveur d'authentification, qui a autorité sur les décisions.
5. Le démon nuauth analyse le paquet reçu (et en particulier l'adresse IP source) et envoie une demande de mise à jour à tous les clients connectés (par l'étape 1) depuis cette adresse pour qu'ils authentifient les connexions en attente. Cette demande est envoyée au travers du (ou des) canal (canaux) crypté(s) ouvert(s) à l'étape 1.
6. Le client qui a ouvert notre connexion " témoin " stipule à nuauth qu'il l'a fait, en précisant tous les paramètres IP (IP et port destination, port source, etc.). À ce stade, le serveur nuauth connaît, de manière sûre, l'identité de l'utilisateur à la source de notre connexion.
7. nuauth réalise une requête sur la base des ACL, pour vérifier si l'utilisateur dispose des

- droits pour établir cette connexion. nauth obtient ainsi une décision.
8. (En option) nauth journalise toutes les informations relatives à cette connexion dans une base SQL, avec bien sûr l'identité de l'utilisateur.
 9. nauth envoie la décision obtenue en 7 à nufw, qui la relaie à son tour à Netfilter. La décision associée à notre utilisateur est alors appliquée par Netfilter.
 10. S'il est autorisé, le paquet poursuit sa route.
 11. (En option) Si le serveur (Apache, par exemple) désire connaître l'identité de l'utilisateur, au lieu de la lui demander directement, il peut réaliser une simple requête SQL SELECT pour récupérer l'identifiant de l'utilisateur en partant de marqueurs uniques de la connexion (la socket source, pour les connaisseurs). L'utilisateur est ainsi authentifié de manière transparente, et sans pouvoir tricher, sur le serveur : il s'agit d'une solution de Single Sign On (authentification unique) très simple, très sûre, et indépendante du protocole.
 12. Le reste du flux est géré par le suivi de connexion (Stateful inspection) de Netfilter : les paquets ne repassent pas par les démons nufw/nauth.



5

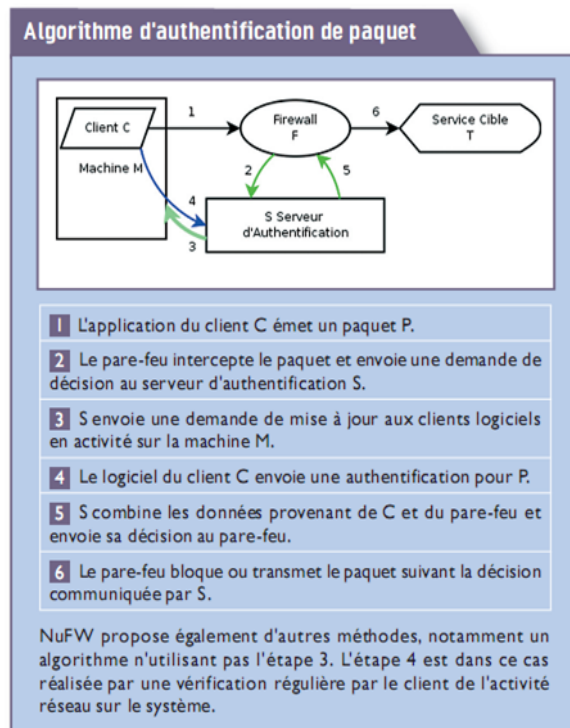
Daemon ufwi-authd

5.1 Introduction

Nuauth command est une interface qui permet de contrôler des fonctions importantes du daemon authd, comme l'obtention de la liste des utilisateurs connectés par exemple. Chaque fois qu'un client envoie un paquet(1) pour commencer une connexion à travers la passerelle, la station cliente envoie un paquet(2) d'identification au daemon authd. Le pare-feu de la passerelle met en file d'attente le paquet et envoie directement des informations au daemon authd.

Le travail du daemon va être d'analyser les deux paquets (1) et (2) et de vérifier si le client a le droit d'initialiser la connexion qu'il demande. Si ufwi-authd indique que le paquet(1) est autorisé alors la connexion est initialisée, sinon la connexion est annulée. Ufwi-authd peut aussi utiliser un serveur LDAP pour la définition des utilisateurs et groupes.

Ci-dessous le un schéma montrant le processus d'authentification utilisé par NuFW, resté inchangé avec UFWI :



5.2 Intallation

Pré-requis :

Script autogen.sh :

- version automake1.7

Compilation Nufw :

- GNU libtool
- GNU make
- libpam-dev
- glib 2.4+
- libipq (iptables-dev pour debian) ou libnetfilter queue
- libldap
- libsasl2
- libgnutls
- libgcrypt

Noyau :

Il est recommandé d'utiliser un noyau récent afin de bénéficier de toutes les dernières nouveautés implémenter dans ce dernier. Une version de noyau supérieur à 2.6.18 est un bon choix. Le patch dump-connection-mark.diff (disponible dans patches/) peut être appliqué au noyau afin d'améliorer les performances de ce dernier lorsque nous utiliserons le log de session.

Compilation :

La compilation du daemon est relativement simple, elle se déroule en quatre étapes :

- Lancement du script ./autogen.sh
- Exécution de ./configure
- make
- Et pour finir make install

Lors de la première installation, il faut penser à copier le fichier de configuration avec la commande suivante :

```
cp ./conf/nuauth.conf /usr/local/etc/nuauth.conf
```

5.3 Quelques commandes liées au daemon

Commandes principales :

- quit : déconnexion
- refresh cache : rafraichit tous les caches
- reload : recharge la configuration du daemon d'authentification

Information :

- help : affiche la liste des commandes utilisables
- version : affiche la version du daemon
- uptime : affiche depuis combien de temp tourne le daemon

Gestion des utilisateurs :

- users : affiche els utilisateurs connectés
- disconnect all : déconnecte tous les utilisateurs
- disconnect ID : déconnecte un utilisateur grâce à son identifiant (ID)

5.4 Fichier de configuration

Le fichier `authd.conf` est le fichier principal de configuration pour le daemon `ufwi-authd`. C'est dans ce fichier que seront indiqués l'adresse du daemon `ufwi-filterd` par exemple ou encore le niveau de debug, le nombre de connexion qu'un utilisateur peut lancer. Dans ce fichier seront aussi renseignés les différents paramètres qui guident le comportement du daemon, mais aussi les paramètres système, et pour finir les chemins absolus des autres fichiers de configuration.

Il existe aussi d'autres fichiers de configurations liés à `ufwi-authd` :

- `modules/nuauth-tls.conf` qui contiendra les paramètres TLS
- `modules/nuauth-krb5.conf` configuration authentification Kerberos 5
- `modules/nuauth-ldap.conf` authentification ldap
- `modules/nuauth-mysql.conf` configuration de la base de données pour les logs utilisateurs (mysql)
- `modules/nuauth-pgsql.conf` configuration de la base de données pour les logs utilisateurs (postgres)

5.5 Installation des certificats

Le daemon d'authentification et le client utilisent des certificats afin de communiquer ensemble. Lors de l'installation de la suite `ufwi` des certificats sont installés par défaut dans le répertoire `/certs`. On peut utiliser ces certificats par défaut pour procéder à des tests mais il n'est pas conseillé de les utiliser pour une utilisation pro de la suite `ufwi`. Dans cette section nous allons voir comment générer ces certificats.

Génère notre propre Autorité de Certification :

```
mkdir private
```

```
chmod 700 private
```

```
openssl req -new -x509 -keyout private/CAkey.pem -out private/CAcert.pem
```

Génère les clés privées pour le client et le daemon d'authentification :

```
openssl genrsa -out private/nufw-key.pem
```

```
openssl genrsa -out private/nuauth-key.pem
```

Génère les demandes de certificats pour le client et le daemon :

```
openssl req -new -key private/nufw-key.pem -out nufw.csr
```

```
openssl req -new -key private/nuauth-key.pem -out nuauth.csr
```

Signe les demandes de certificats grâce à l'AC :

```
openssl x509 -req -days 365 -in nufw.csr -CA private/CAcert.pem -CAkey private/CAkey.pem -CAcreateserial -out nufw-cert.pem
```

```
openssl x509 -req -days 365 -in nuauth.csr -CA private/CAcert.pem -CAkey private/CAkey.pem -CAcreateserial -out nuauth-cert.pem
```

Enfin on déplace les certificats dans le bon répertoire :

```
cp private/nufw-key.pem /etc/nufw/
```

```
cp nufw-cert.pem /etc/nufw/
```

```
cp private/nuauth-key.pem /etc/nufw/
```

```
cp nuauth-cert.pem /etc/nufw/
```

5.6 Configuration basique du daemon

Lors de l'installation de la suite `ufwi` un fichier de configuration de base est fournie `nuauth.conf`, qui est disponible dans le répertoire `conf`. Ce fichier est remplie de plusieurs directives, les deux plus importantes directives de configuration sont : `nuauth client listen addr` : qui définit l'adresse à laquelle le daemon va attendre les requêtes des clients `nuauth nufw listen addr` : qui définit l'adresse à laquelle le daemon va attendre les requêtes de `nufw`. La liste des machines autorisées à se connecter au serveur d'authentification constitue la variable `nufw gw addr`.

Ensuite, on doit choisir le module d'authentification et de vérification des ACL. Les modules suivants sont disponibles :

- `libldap` : les informations utilisateur sont stockées dans un annuaire LDAP
- `dbm` : les informations utilisateur sont stockées dans une base `gdbm`
- `plaintext` : les informations utilisateur sont stockées dans un fichier texte
- `system` : l'authentification s'adosse à PAM et utilise les groupes existants dans le système.

Ceci procure un moyen pratique d'utiliser `nss` et/ou `pam-modules`

Le module d'authentification est paramétrable via l'option `nuauth user check module` dont la valeur par défaut est `libsystem` (si non défini dans le fichier de configuration). D'autre paramètres concernant la vérification des ACL doivent être précisés si on choisit une des deux authentification suivante :

- `libldap`
- `plaintext`

Tout ceci en définissant la variable `nuauth acl check module`.

6

Daemon ufwi-filterd

6.1 Introduction

Le daemon ufwi-filterd (anciennement appelé nufw) n'est d'autre qu'un pare-feu basé sur NFQUEUE netfilter. Il permet d'écrire des règles de filtrage basées sur l'identité des utilisateurs, en plus des critères de réseau classiques. L'authentification est effectuée de façon transparente en requérant les informations d'identification de l'utilisateur avant toute quelconque décision de filtrage ne soit prise. En pratique, cela signifie que les politiques de filtrage peuvent intégrer l'annuaire utilisateur, et amène cette notion d'ID utilisateur au niveau de la couche IP.

Ufwi-filterd est capable de :

- Filtrer le trafic en fonction du système d'exploitation et des applications utilisées par les utilisateurs distants.
- marquer chaque paquet d'une connexion avec l'identifiant de son utilisateur et donc d'appliquer une politique de qualité de service spécifique à chaque utilisateur.
- contribue de manière très pointue à la surveillance de l'activité réseau des serveurs.
- dispose de modules de surveillance qui journalisent les événements principaux de l'activité du réseau en indiquant quels sont les utilisateurs à l'origine des flux.

6.2 Installation

Une installation typique de la suite logicielle NuFW comporte 2 démons : nufw (ufwi-filterd) et nuauth (ufwi-authd) et autant de clients que nécessaire.

Pré-requis :

- automake1.7 pour exécuter autogen.sh
- GNU libtool
- GNU make

Pré-requis pour la compilation et l'exécution de ufwi-filterd :

- ufwi-base
- ufwi-confparser
- ufwi-ssl

Il est recommandé d'utiliser un noyau récent afin de bénéficier de toutes les dernières nouveautés implémentées dans ce dernier. Une version de noyau supérieure à 2.6.18 est un bon choix.

6.3 Compilation

La compilation de ufwi-filterd est relativement simple elle se resume a utiliser les commandes suivantes :

- ./autogen.sh
- ./configure
- make
- makeinstall

Lors de la première installation, il ne faut pas oublier de copier le fichier de configuration "make install-conf" afin de chercher les changements entre votre fichier de conf actuelle et le nouveau.

Un fichier INSTALL avec toutes les instructions a suivre est fournie dans le dossier de ufwi-filerd disponible a partir de se lien [http ://ufwi.org/projects/ufwi-filterd/ repository](http://ufwi.org/projects/ufwi-filterd/repository) .

6.4 Commandes

Tout dabord, vous devez executer en root ufwi-filterd. ufwi-filterd -h vous donnera un message d'aide pour l'utilisation de ufwi-filterd.

6.5 Fichier de configuration

Le fichier de configuration de ufwi-filterd se nome tout simplement "filterd.conf". On pourra le trouver dans /etc/ufwi-filterd/. Dans se fichier on trouvera l'adresse ou le nom du serveur d'authentification nuauth (par default 127.0.0.1), on trouvera aussi les chemin absolu des fichiers :

- /etc/ufwi-filterd/key.pem (clé privé du serveur)
- /etc/ufwi-filterd/cert.pem (certificat du serveur)
- /etc/ufwi-filterd/cacert.pem
- /etc/ufwi-filterd/crl.pem (liste de révocation de certificat serveur)

7

Daemon ufwi-rcpd

7.1 Introduction

Le module ufwi-rcpd (anciennement appelé NuCentral) est le module qui gère les autres daemons.

7.2 Installation

7.2.1 Pré-requis

Avant de lancer l'installation du module, certains pré-requis sont nécessaires :

- Python 2.5
- Twisted web
- M2Crypto
- Jinja
- Subversion (svnadmin program)
- sudo
- pysvn
- pytz

Debian :

```
apt-get install python-twisted-web python-svn  
python-m2crypto sudo python-jinja subversion python-tz
```

Pour l'installation de ufwi-rcpd :

- (GNU) make
- sqlite3

Sous Debian :

```
apt-get install make sqlite3
```

Paquets optionnels : Pour compiler les fichiers de .ts à .qm et mettre à jours les transmissions :

- lrelease4 : Qt development tools
- pylupdate4, lrelease-qt4 : Python Qt development tools

Sous Debian :

```
apt-get install libqt4-dev pyqt4-dev-tools
```

Autres :

- python-twisted-snmp : pour le module SNMP.
- gnutls-bin : le programme certtool l'utilise pour générer les certificats ssl.

- `pytest` (Sous Debian, `python-codespeak-lib`) : utilisé pour les tests.
- `libconfig-inifiles-perl` : pour `tools/ufwi_rpcd_enmod`.
- `IPy` (`python-ipy`) : pour les tests.

7.3 Utilisation

Une fois démarrer, on peut accéder au fonction plus particulaire de `ufwi-rpcd` (`nucentral`).

7.3.1 Démarrage

Lancement en mode daemon

Pour lancer `ufwi-rpcd` en mode daemon il suffit d'exécuter une commande en root :

```
twistd -y /usr/sbin/ufwi-rpcd.tac --pidfile=/var/run/ufwi-rpcd.pid  
-l /var/log/ufwi-rpcd-twisted.log
```

Pour arrêter `ufwi-rpcd` en mode daemon (en root) :

```
kill $(cat /var/run/ufwi-rpcd.pid)
```

Lancement en mode "premier plan"

Pour facilité le développement, on peut également lancer `ufwi-rpcd` en premier plan :

```
twistd -n -y /usr/sbin/ufwi-rpcd.tac -l /var/log/ufwi-rpcd-twisted.log
```

Pour arrêter le processus : CTRL+c.

7.3.2 `ufwi_rpcd_client` (`nucentral_client`)

Le client

Pour lancer le client d'`ufwi_rpcd` en HTTP (tcp/8080) :

```
ufwi_rpcd_client --host 127.0.0.1 --cleartext -u admin -p admin
```

Par défaut, `ufwi_rpcd_client` utilise HTTPS (tcp/8443) :

```
ufwi_rpcd_client --host 127.0.0.1 -u admin -p admin
```

Une fois connecté sur le client, le prompt devient :

```
nucentral>
```

Le menu

Une fois lancer, on peut accéder aux différentes fonctions d'`ufwi-rpcd` :

- `call('component', 'service', ...)`
Appelle un service d'`NuCentral`
- `authenticate('login', 'password')`
Authentification ou mise à jour d'un groupe ou d'un utilisateur.
- `components()`
Affiche la liste des différents composants.
- `services('component')`
Affiche la liste des services d'un composant.
- `proxy('component')`
Crée un composant pour le proxy qui pourra être utilisé avec `component.service()`.

- `nucentralStatus()`
Afficher l'état du serveur et la session d'informations.
- `help('component')`
Utilisation d'un composant.
- `help('component', 'service')`
Utilisation d'un service d'un composant.
- `help(proxy)`
Utilisation du proxy.
- `exit()`
Quitter.
- `pyhelp(object)`
Aide Python pour l'objet spécifié.

Différents composants sont disponibles :

`CORE, access, acl, audit, auth_cert, bind, config, contact, hostname, hosts, httpout, localfw, lock, logger, network, ntp, nuauth, nuauth_command, nuconf, nuface, nalog, nupki, nurestore, reporting, resolv, session, status, streaming, system, system_info, tools, update, users_config, versionning`

8

Avancement du projet à mis parcours

Ce qui a été fait :

- Ufwi est installé et fonctionnel.
- La première partie du projet est terminer. Il reste les tests de fonctionnement à faire

Ce qui reste a faire :

- Recherche des limites et des failles du logiciel
- Comment gérer l'élévation de privilèges (root se connect en ssh sur un non-root, que se passe t-il ?)
- Mettre en lumière les durées de vie des autorisations données.
- Comment s'assurer que l'utilisateur est encore présent ?
- Ufwi aide t-il pour l'authentification des utilisateurs nomades connu (commercial) ou non (intervenant exterieur) ?

9

Difficulté rencontré et avis personnel